TK 90X

manual de operação

MICRODIGITAL

Manual de Operação

TK90X

ÍNDICE

INTRODUÇÃO I-7

APRESENTAÇÃO I-8

INSTALANDO SEU TK90X I-9

1. TECLADO 1-1
1.1. Modos de Comandar o Teclado 1-1
1.2. DELETE 1-2
1.3. ENTER 1-4

2. DISPOSIÇÃO DA TELA DE TV 2-1
2.1. Margem Inferior 2-1

3. O TK90X COMO CALCULADORA 3-1
3.1. Modo Imediato 3-1
3.2. Os Cálculos 3-2
3.2.1. Potenciação 3-3
3.2.2. Radiciação 3-3
3.2.3. Raiz Quadrada 3-3
3.3. Expressões Algébricas 3-3

4. OS NÚMEROS NO TK90X 4-1
4.1. Notação Científica 4-1
4.2. Limites Numéricos 4-1
4.3. Forma dos Números 4-1

5. PROGRAMAÇÃO BASIC 5-1
5.1. Conceitos 5-1
Os Dados 5-1
Variáveis 5-1
5.2. Comandos: 5-1
PRINT 5-1
LET 5-2
EDIT 5-3
DELETE 5-4
RUN 5-5
CLS 5-6
LIST 5-6
AT 5-8
TAB 5-10
NEW 5-11
REM 5-11
: (dois pontos) 5-11
GOTO 5-12
INPUT 5-12
STOP 5-13
CONT 5-13

6. DECISÕES 6-1
6.1. IF..THEN...
STOP 6-1
6.2. Comparando Dados 6-2
6.2.1 Comparações Numéricas 6-2

6.2.2 Comparações Alfanuméricas 6-3

7. INSTRUÇÃO FOR...NEXT 7-1
7.1. Uso de FOR...NEXT como Artifício Matemático 7-2
7.2. STEP 7-3

8. SUB-ROTINAS 8-1
GOSUB...RETURN 8-1

9. MODO TRACE: RECURSO DE DEPURAÇÃO 9-1

10. TABELAS DE DADOS 10-1
10.1. READ...DATA 10-1
10.2. RESTORE 10-2

11. COMO DENOMINAR VARIÁVEIS 11-1

12. MANIPULANDO STRINGS 12-1
12.1. Determinando um "corte" 12-1
12.2. Mesclando Strings 12-2
12.3. Somando Strings 12-3
12.4. Manipulando o Comprimento do String - LEN 12-4

13. FUNÇÕES 13-1
13.1. STR$ 13-1
13.2. VAL 13-1
13.3. VAL$ 13-2
13.4. DEF FN 13-3
13.5. SGN 13-5
13.6. ABS 13-5
13.7. INT 13-5

14. FUNÇÕES MATEMÁTICAS 14-1
14.1. EXP 14-1
14.2. LN 14-1
14.3. PI 14-1
14.4. Funções Trigonométricas 14-2

15. NÚMEROS RANDÔMICOS 15-1
15.1. RND 15-1
15.2. RAND 15-1

16. MATRIZES 16-1
16.1. DIM 16-1
16.2. Matrizes Multidimensionais 16-2
16.3. Matrizes Tridimensionais 16-3

17. CORES 17-1
17.1. INK e PAPER 17-1
17.2. BRIGHT 17-2
17.3. FLASH 17-3
17.4. BORDER 17-3
17.5. INVERSE 17-3
17.6. OVER 17-6
17.7. ATTR 17-6
17.8. INV. VIDEO E TRUE VIDEO 17-7

18. O CONJUNTO DE CARACTERES 18-1
18.1. CHR$ 18-1
18.2. CODE 18-1
18.3. Caracteres do Modo Gráfico 18-2
18.4. UDG 18-2
18.5. Definindo Caracteres 18-3
18.6. SCREENS$ 18-7

19. MANIPULANDO A MEMÓRIA 19-1
19.1. POKE 19-1
19.2. PEEK 19-1
19.3. BIN 19-1

20. GRÁFICOS 20-1
20.1. PLOT 20-1
20.2. DRAW x,y 20-1
20.2.1. DRAW x,y,z 20-2
20.3. POINT x,y 20-2
20.4. CIRCLE x,y,r 20-3

21. TEMPO E MOVIMENTO 21-1
21.1. PAUSE 21-1
21.2. INKEY$ 21-2

22. EFEITOS SONOROS 22-1

23. LEITURA E GRAVAÇÃO DE FITAS 23-1
23.1. EASY-LOAD 23-1
23.2. SAVE 23-1
23.3. LOAD 23-2
23.4. SAVE...LINE 23-3
23.5. SAVE...DATA 23-3
23.6. VERIFY...DATA 23-4
23.7. LOAD...DATA 23-4
23.8. SAVE...CODE 23-5
23.9. LOAD...CODE 23-6
23.10. LOAD...SCREEN$ 23-6
23.11. MERGE 23-6
23.12. TABELA DE INSTRUÇÕES 23-8

24. MEMÓRIA 24-1
24.1. Tipos de Memória 24-1
24.2. Mapeamento da RAM 24-1
24.2.1. Arquivo de Imagem 24-2
24.2.2. Atributos 24-3
24.2.3. Buffer da Impressora 24-3
24.2.4. Variáveis do Sistema 24-3
24.2.5. Mapa do Speed-drive 24-3
24.2.6. Informacão do Canal 24-3
24.2.7. Área do Sistema BASIC 24-3
24.2.8. Pilha de Cálculo 24-4
24.2.9. Pilha de Máquina 24-4
24.2.10. Pilha de GOSUB 24-4
24.2.11. Definição de Gráficos 24-4
24.3. Linha de Programa em Memória 24-4
24.4. RAMTOP 24-4

24.5. CLEAR 24-5

25. CÓDIGO DE MÁQUINA 25-1
25.1. Código Binário 25-1
25.2. USR 25-2
25.3. IN-OUT 25-3

26. PERIFÉRICOS 26-1

APÊNDICES

A. Programas A1

B. Mensagens B1

C. Variáveis do Sistema C1

D. Código de Caracteres D1

INTRODUÇÃO

A você, Prezado Usuário, destina-se este manual. Trata-se de um livro de leitura leve e de acompanhamento prático e fácil, livre de complicações técnicas.

Compõe-se de de vinte e seis capítulos, quatro apêndices contendo programas, mensagens, código de caracteres, variáveis do sistema; e ainda uma seção destinada à instalação e à detecção de eventuais problemas de funcionamento.

Este manual irá orientá-lo para que você conheça a linguagem BASIC do TK90X. A partir desta orientação, você conseguirá dominar os princípios de programação. Então, já estará apto para começar a explorar todos os recursos de seu computador. Siga corretamente as instruções aqui dadas, de maneira a conhecer sua máquina e saber como se comunicar com ela. Quando achar que pode praticar o que aprendeu e criar seus próprios programas, faça-o.

Procure exercitar ao máximo seu aprendizado fazendo tentativas, reformulando os exemplos, mudando variáveis, até satisfazer sua curiosidade (ou despertá-la ainda mais) e obter respostas para as suas dúvidas. Quando estiver manejando razoavelmente a linguagem e o aparelho, notará que se trata de uma máquina extremamente simples de operar. Verá também que ela pode ser tanto uma ferramenta de trabalho muito precisa, como uma divertida fonte de lazer.

Vá em frente, conheça seu TK90X e comande-o.

APRESENTAÇÃO

O TK90X emprega uma linguagem muito difundida entre os microcomputadores pela sua simplicidade e versatilidade, o BASIC. Seu computador, no entanto, possui comandos exclusivos, tais como: INK, PAPER, BORDER, BRIGHT, UDG, SOUND e TRACE.

Através do comando INK, você determina a cor dos caracteres que aparecerão na tela ou a cor do traçado que pretende criar.

Com a palavra-chave PAPER, você escolhe a cor de fundo da tela.

Por meio do comando BORDER, você seleciona a cor das margens da tela, indepedentemente da cor central.

E mais ainda, com a instrução BRIGHT, você realça as oitos cores disponíveis no TK90X, tornando mais intenso o colorido de suas produções.

O seu TK90X resolveu aquela situação restritiva de somente se poder escrever com maiúsculas e sem acentuação gráfica. Ele é capaz de gerar o alfabeto inteiro em tipos minúsculos, maiúsculos e ainda possui dois conjuntos com os principais caracteres acentuados para os idiomas, português e espanhol, acessados pela função UDG. Esta função permite, também, que você use sua criatividade para gerar novos caracteres e aplicá-los em seus programas.

Se você resolver criar músicas ou efeitos sonoros em programas, sinta-se à vontade. Seu micro dispõe do comando SOUND (sintetizador de som controlado por software) que, além das opções de controle da tonalidade e do tempo de duração das notas, ainda envia os sinais sonoros para o alto-falante de sua TV.

Para facilitar o interfaceamento de seu micro com gravador, foi adotado o sistema "easy-load" (carregamento fácil). Este sistema, assegura maior eficiência na leitura e gravação de fitas magnéticas, além de apresentar, em seu vídeo, um "feedback" do que está ocorrendo durante a operação de carga.

Tão útil ou mais, é a possibilidade oferecida pelo modo TRACE de acompanhar-se a evolução do computador na execução de um programa BASIC. Esta condição ajuda a detectar erros lógicos em seu programa.

Finalmente, outra característica do TK90X digna de nota é a possibilidade da inserção de vários comandos numa mesma linha de programa. Para tanto, basta separá-los por dois pontos. Este recurso é bastante utilizado em computadores de maior porte, o que evidencia que o TK90X é uma máquina compacta e poderosa.

INSTALANDO SEU TK90X

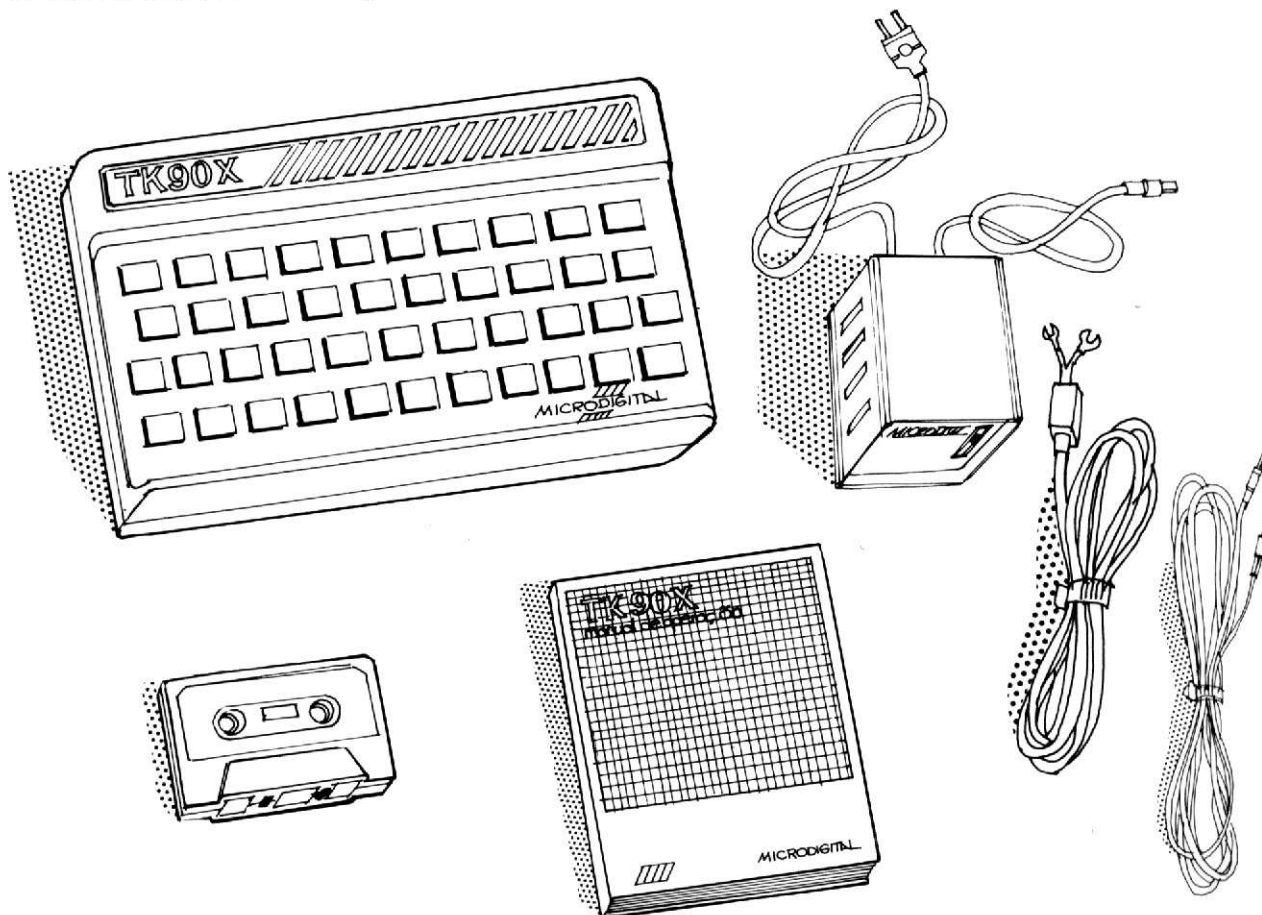
Para deixar seu TK9ØX pronto para entrar em ação, você deve verificar se tem à mão todos os artigos da "LISTA DE ITENS" que se encontra mais adiante. Depois, basta seguir as instruções de instalação.

Faça as conexões de maneira que fiquem firmes. Lembre-se de que, se acidentalmente interromper-se a alimentação de energia pelo desligamento de um cabo, você perderá seu programa e todas as informações, e resultados.

Quando o computador não estiver em uso, desligue a Fonte de Alimentação e desconecte-a da tomada.

LISTA DE ITENS

Ao abrir sua embalagem você encontrará:

1. Seu microcomputador TK9ØX
2. A FONTE DE ALIMENTAÇÃO de corrente contínua necessária ao funcionamento do TK9ØX
3. O CABO DE VÍDEO, que serve para ligar o computador ao aparelho de TV
4. O CABO DE GRAVADOR para leitura/gravação que liga seu TK9ØX a um gravador cassete comum
5. A FITA ARCO-ÍRIS contendo instruções e programas aplicados ao TK9ØX
6. Este MANUAL DE OPERAÇÃO com o Cartão de Referência

Você deverá providenciar:

1. um aparelho de TV
2. um gravador cassete de boa qualidade

TIRE AS DÚVIDAS:

— A TV deve ser em cores?

Não. Porém, é claro, você não poderá obter, numa TV em branco e preto, as cores que o TK90X pode produzir.

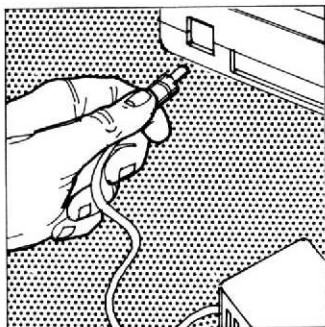
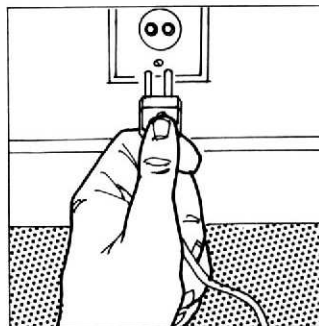
— Qual é a alimentação elétrica correta?

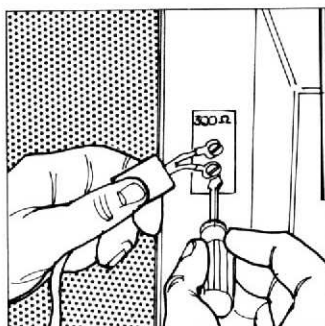
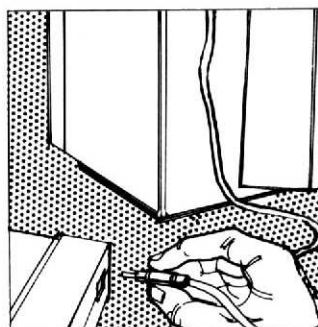
O TK90X utiliza-se da rede elétrica local tanto de 110 como de 220 V AC (verifique se a Fonte de Alimentação está adequada à voltagem da rede local). Em redes onde há grande oscilação de voltagem, haverá a necessidade de um estabilizador. Consulte o seu Serviço Técnico.

— O TK90X produz interferência?

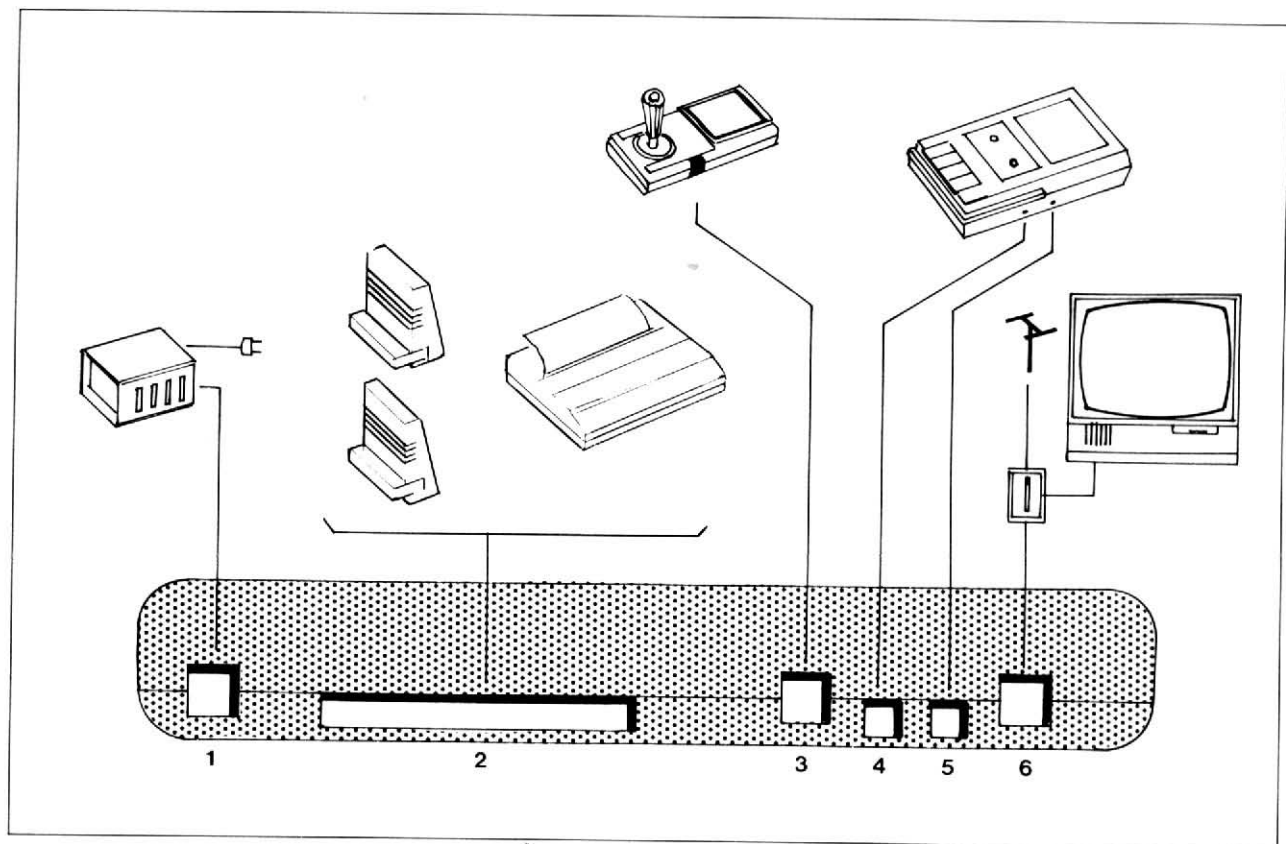
O TK90X pode interferir num rádio colocado muito próximo dele. Mas isto não danifica nem o micro e nem o rádio.

LIGANDO O TK90X À CORRENTE ELÉTRICA

Se você já certificou-se de que está com a Fonte de Alimentação adequada e posicionada na voltagem da rede elétrica local, siga a seqüência de instruções que as ilustrações apresentam:

Introduza o pequeno plugue do fio que sai da Fonte de Alimentação, na tomada DC, situada na parte traseira do TK90X.

Ligue o plugue do outro fio da Fonte numa tomada AC. Posicione a chave liga/desliga da Fonte de Alimentação em OFF

LIGANDO O TK90X AO VÍDEO.

Desconecte a antena da TV e, no lugar dela, instale o CABO de VÍDEO.

Introduza o outro plugue do cabo conector de vídeo na tomada VIDEO do computador.

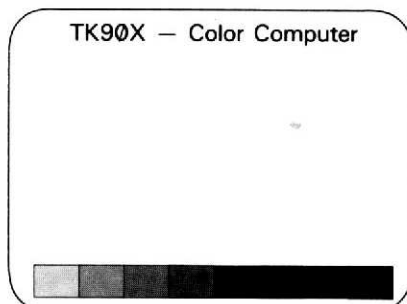
Ligue o aparelho de TV no canal 3. Agora está tudo pronto para que você dê início à sintonização do televisor, de maneira que capte os sinais do TK90X. Posicione a chave da Fonte de Alimentação em ON.

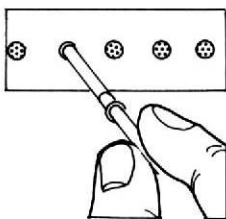
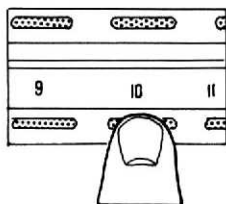
AS TOMADAS E CONEXÕES DO TK90X

1. DC - tomada de Energia serve para conectar o plugue da Fonte de Alimentação
2. EXPANSION - Encaixe de dispositivos periféricos diversos
3. JOYSTICK - Conexão dos comandos de jogo
4. MIC - Os sinais que saem por esta conexão são levados pelo cabo de gravação à tomada MIC do gravador
5. EAR - Conecta o cabo de leitura da tomada EAR do gravador para transferir sinais deste para o computador
6. VIDEO - Recebe o plugue do cabo conector de vídeo, para transmitir sinais de áudio e vídeo para a TV

Nota: O seu TK9ØX está preparado para receber acessórios compatíveis, existentes no mercado. Podem, existir, eventualmente, periféricos com consumo de energia total superior à fornecida pela Fonte de Alimentação normal.
Se você pretende conectar acessórios no seu TK9ØX, consulte seu Serviço Técnico para saber que tipo de Fonte de Alimentação deve ser usada.

SINTONIZANDO A TV

O TK90X gera sinais de vídeo para TV em cores, bem como sinais sonoros. A TV deve ser ajustada na freqüência do canal 3 (mesmo que o aparelho seja em B&P).

Se a sintonia da freqüência da TV estiver captando os sinais do computador, você deverá obter o seguinte display de apresentação:

Caso a imagem não tenha sido obtida, você deve ajustar a sintonia fina do televisor.

SELETOR TRADICIONAL
Geralmente o ajuste de sintonia encontra-se acoplado ao próprio seletor. Gire-o em um determinado sentido até obter imagem.
Se for necessário, inverta o sentido, e tente novamente.

SISTEMA PUSH-BOTTON
Neste sistema, o ajuste da sintonia fina é independente para cada canal e está localizado fora do seletor de canais.
Procure o botão correspondente ao canal 3 e gire-o respectivamente para ambos os lados até obter o display de apresentação.

COMO AJUSTAR AS CORES DO TK90X

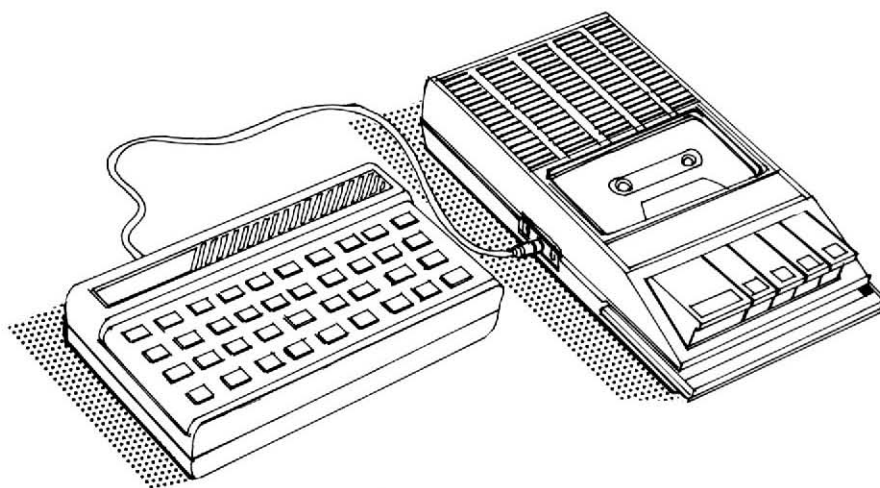
Depois de haver sintonizado a TV de maneira a receber com nitidez os sinais de imagem gerados pelo computador, você pode ajustar as cores. Este ajuste se faz pela faixa inferior do display de apresentação.

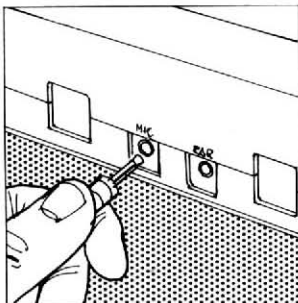
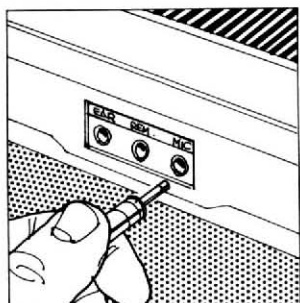
Se você possui TV em cores, verá no display as oito cores que o TK9ØX produz:

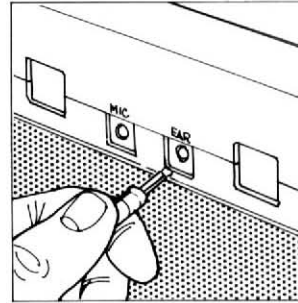
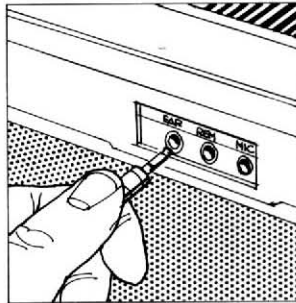
branco	amarelo	ciano	verde	magenta	vermelho	azul	preto

Procure os controles de cor, brilho e contraste do seu televisor e ajuste-os até obter a melhor apresentação. É bom lembrar que a tonalidade das cores pode variar de televisor para televisor. Se seu televisor for em branco e preto, será apresentada uma escala de tonalidades que variam do cinza claro ao preto.

COMO CONECTAR O GRAVADOR CASSETE

O cabo de leitura e gravação que acompanha o TK9ØX serve para conectar o computador a um gravador, permitindo assim que você armazene (SAVE) informações em fita magnética ou obtenha da fita informações previamente armazenadas (LOAD).

PARA SALVAR (SAVE) INFORMAÇÕES

Introduza um dos plugues do cabo de leitura e gravação na tomada MIC do gravador e o outro na tomada MIC do TK9ØX. (VEJA CAPÍTULO 23).

PARA CARREGAR (LOAD) INFORMAÇÕES

Introduza um dos plugues do cabo de leitura e gravação na tomada EAR do gravador, e o outro, na tomada EAR do TK9ØX. (VEJA CAPÍTULO 23).

DEFEITOS E CAUSAS PROVÁVEIS

Se você tiver problemas durante a operação do seu TK90X, verifique a seguinte tabela de sintomas. Se não conseguir solucionar o problema, procure a assistência técnica autorizada.

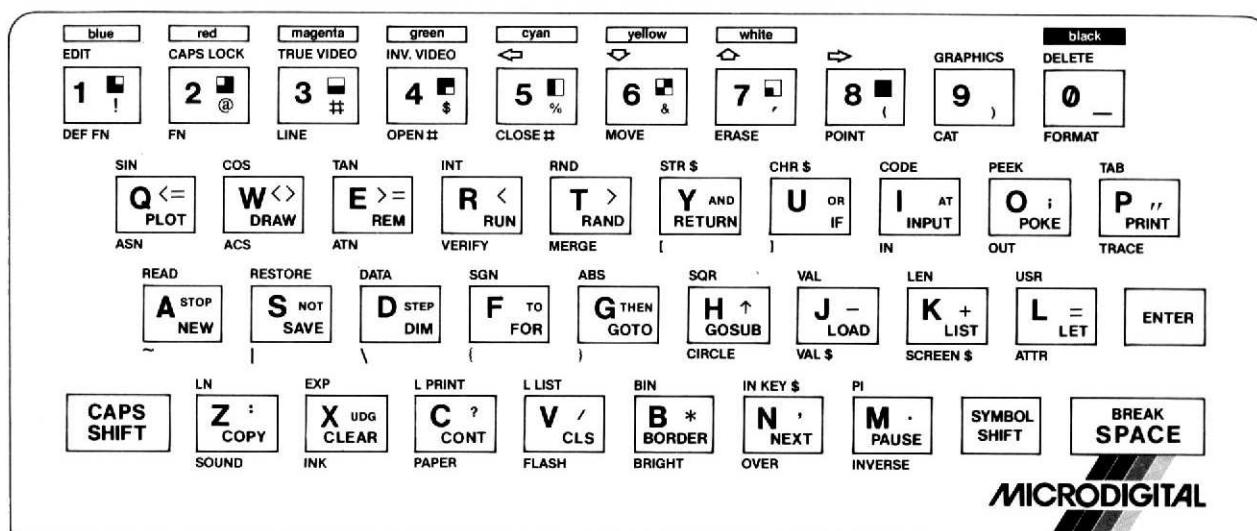
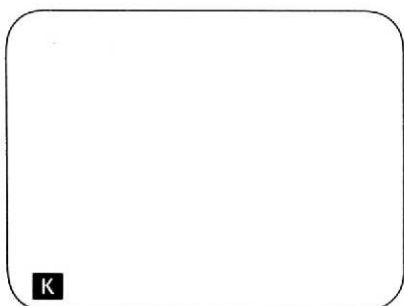
SINTOMA	DIAGNÓSTICO
O Display de Apresentação não aparece quando você liga o computador:	1. Não há alimentação de corrente alternada. Verifique a conexão do plugue à tomada DC. 2. Não foi obedecida a seqüência para ligar o computador e seus acessórios. Todos os acessórios devem ser conectados antes de se ligar o computador. 3. Acessório não conectado corretamente. Verifique a conexão. 4. Sua televisão não está ajustada. Verifique o contraste, o brilho e a sintonia fina. 5. O cabo de vídeo não foi conectado corretamente. Conecte-o conforme a instrução.
Você não consegue carregar seu programa em fita cassete:	1. A conexão do cassete não foi feita corretamente. Verifique as instruções de conexão do gravador. 2. O volume do seu gravador está muito baixo. Neste caso as faixas indicadoras de carga (BORDER) não aparecerão. Aumente o volume de seu gravador. 3. O cabeçote de seu gravador pode estar desalinhado. Para alinhá-lo, coloque uma fita de programas no gravador, desconecte a tomada EAR e pressione "PLAY" (ou tecla correspondente), no gravador. Com uma ferramenta apropriada (chave de fenda ou PHILIPS), movimente o parafuso de regulagem de cabeçote para a direita ou esquerda, até obter o som mais estridente e alto possível. Este parafuso fica geralmente do lado esquerdo do cabeçote de gravação/leitura de seu gravador.
O computador "trava" durante sua operação sendo necessário desligar o aparelho:	1. Houve flutuação de energia. Verifique se a alimentação está correta. 2. Conector da fonte com defeito ou instalado incorretamente. Verifique se todos os cabos de conexão estão conectados ou se eles não estão danificados. 3. Erro de programação. Reveja o programa.

TECLADO 1

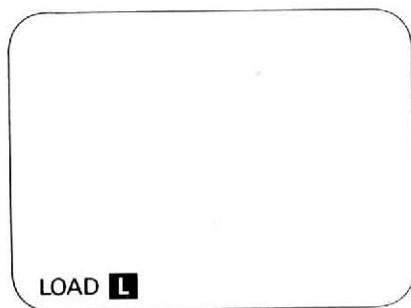
1. TECLADO

O teclado é a via pela qual você registra caracteres para a entrada de dados na tela e na memória do computador, e é o meio através do qual você solicita tarefas de processamento ao aparelho.

1.1. MODOS DE COMANDAR O TECLADO

O teclado do TK90X compreende caracteres alfabéticos maiúsculos e minúsculos, caracteres numéricos, caracteres simbólicos, caracteres gráficos, comandos e funções. Os comandos (palavras-chaves) são acessados ao pressionarem-se as teclas nas quais estão inscritos. Ex.: POKE, PRINT, LIST são comandos diretos do teclado. Não há necessidade de digitá-los letra por letra.

O TK90X opera em diversos modos, que são indicados pelo cursor. O cursor, representado inicialmente por uma letra K, no canto inferior esquerdo da tela, informa que o computador está à espera de um comando. Ele mostra também a posição do próximo caractere a ser digitado.

As palavras-chaves aparecem sempre no início de uma linha de programa ou após o comando THEN.

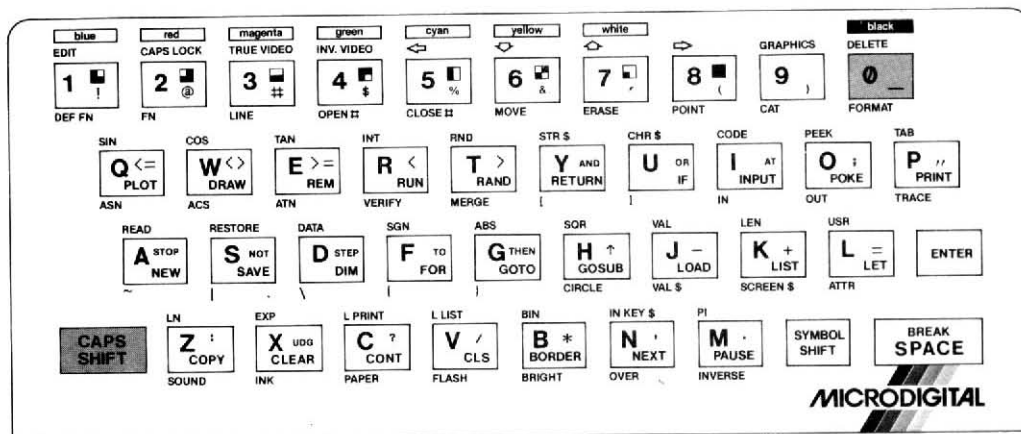
Vamos usar o comando LOAD (contido na letra J) como exemplo e ver o que acontece.

Repare que após ter sido inserido um comando, o cursor muda automaticamente para L (agora à direita da palavra-chave que você pressionou).

A letra L significa que o computador está à espera de um caractere e não de um comando ou função. Digite qualquer coisa, para observar o comportamento do teclado.

Experimente manter uma tecla qualquer pressionada por algum tempo. Você vai notar que o seu TK9ØX é dotado de um repetidor automático de caracteres. Com o tempo você verá como isto facilitará o seu trabalho de digitação.

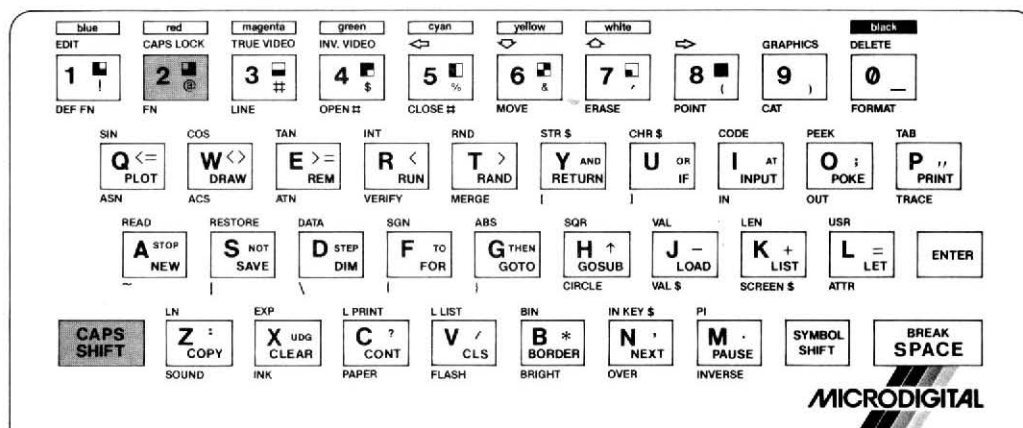
Depois destas experiências com o teclado, sua tela deve estar repleta de caracteres sem significado. Está na hora de você utilizar outro modo de comando.

1.2. DELETE

Pressione a tecla <CAPS SHIFT> em conjunto com a tecla <Ø> (zero).

Mantenha as duas pressionadas até limpar toda a tela. Você utilizou a função DELETE, inscrita em branco acima da tecla <Ø>.

Mantendo a tecla <CAPS SHIFT> pressionada, digite caracteres alfabéticos (de A a Z): você está produzindo caracteres maiúsculos.

Solte a tecla <CAPS SHIFT> e os caracteres aparecerão minúsculos, a exemplo do que ocorre numa máquina de escrever.

Com a tecla <CAPS SHIFT> pressionada, acione <CAPS LOCK>, a função contida na parte superior da tecla <2>.

Digite novamente qualquer letra de A a Z. Não se importe se a primeira letra se transformar num comando, mas preste atenção às outras e ao cursor. Ele agora está no modo **C** e na tela os caracteres são impressos em caixa alta (maiúsculas).

Nota: Para desativar qualquer tecla de alteração de modo, repita a mesma seqüência que você digitou para ativar.

É importante observar que <CAPS LOCK> acionada não dá acesso às funções inscritas em branco acima das teclas da primeira fileira (1-0). Para usá-las é sempre necessário pressionar <CAPS SHIFT>. Apague tudo usando DELETE ou desligando e religando o computador.

Recomece o exercício. Repare que na maioria das teclas existem palavras-chaves e símbolos inscritos em vermelho. Para gerá-los é necessário pressionar a tecla <SYMBOL SHIFT>, localizada no canto inferior direito do teclado.

Digite, por exemplo, 2 + 2 ou TK90X. Para gerar espaços entre os caracteres, use a tecla <BREAK SPACE>. Pressione-a de maneira simples e direta para que ela funcione de modo similar ao da barra de espaço de uma máquina de escrever.

Inscritas em azul e acima de cada tecla ou em vermelho e abaixo delas, encontram-se várias outras funções. Essas funções pertencem ao modo extendido de comando, e o cursor para representá-lo usa a letra **E**.

Para entrar no modo extendido **E**, pressiona-se <CAPS SHIFT> juntamente com <SYMBOL SHIFT>. As funções em azul são obtidas de maneira direta e as representadas em vermelho, com o auxílio de <SYMBOL SHIFT>.

Observe que as combinações acima descritas somente funcionam para a entrada de uma função por vez. Após a entrada da função, seja inscrita em azul ou em vermelho, o cursor voltará automaticamente ao modo **L**.

O modo gráfico é acionado por <CAPS SHIFT> e <9>. Permanece ativado até que <9> seja novamente pressionada. O cursor usa **G** para representar este modo. Há oito caracteres gráficos, contidos nas teclas numéricas (1-8). Outros caracteres gráficos podem ser definidos pelo usuário, como será visto bem mais adiante.

É necessário saber que, para utilizar o computador de maneira programada e não no modo direto, devem-se organizar as instruções numa seqüência lógica e dentro de linhas numeradas em ordem crescente.

Quando o computador está mostrando o cursor **K**, além de comandos do tipo palavra-chave, ele aceita também enumeração de linhas de programação.

Após o número da linha ter sido digitado, o cursor permanece em **K**, para a entrada de uma palavra-chave.

Faça uma experiência, digitando primeiramente o número 10. Logo a seguir tecle a letra P. Deverá aparecer o comando PRINT na tela. Depois do comando, o cursor passa automaticamente para **L**.

1.3 ENTER

A tecla <ENTER> indica ao computador que a linha de comando ou instrução de processamento foi terminada. Depois que você pressionar <ENTER>, o TK90X examinará o que foi escrito. Se a instrução for intelegível para o computador, será aceita; se houver algo incompreensível, aparecerá, no lugar do erro, um sinal de interrogação (**?**), e, eventualmente, uma mensagem de erro. Se isto acontecer, a linha de comando deverá ser corrigida ou apagada.

O sinal **?** pode indicar também a omissão de algum caractere necessário à instrução.

Nota: Os sinais <e> são utilizados ao longo deste manual para simbolizar uma tecla, não devendo, neste caso, serem digitados. Numa instrução do tipo tecle <ENTER>, significa que você deve pressionar apenas a tecla ENTER.

DISPOSIÇÃO DA TELA DE TV

2

2. DISPOSIÇÃO DA TELA DE TV

A tela tem 24 linhas de 32 colunas.

Os comandos introduzidos aparecem na margem inferior do vídeo. Quando uma linha de comando é editada, e você tecla <ENTER> para enviá-la ao computador, ela passa da margem inferior para a parte superior da tela.

O computador irá dispor cada instrução numa listagem ordenada, sempre em ordem crescente.

O resultado das operações, bem como a saída dos programas, aparecem sempre na parte superior do vídeo, a partir da última linha ocupada, se esta não for apagada.

Quando a tela é ocupada até a 22ª linha, o computador se encarrega de mover toda a imagem uma linha para cima (scroll), não alterando a margem inferior. Se este processo implicar no desaparecimento de uma linha que você pode não ter tido a oportunidade de ver, o computador interrompe-o, perguntando se você autoriza que ele mova o conteúdo da tela. A pergunta aparece como "scroll?", no canto inferior esquerdo do vídeo. Sendo a resposta <N>, <SPACE> ou <STOP>, o programa ou qualquer atividade na tela serão interrompidos. Na borda inferior aparecerá a mensagem "D BREAK - CONT repete, 0:1". Não sendo a resposta nenhuma dessas três teclas, o scroll se efetuará.

2.1. MARGEM INFERIOR

A margem inferior compreende duas linhas e é reservada para a entrada de dados (INPUT), de linhas de programa, mensagens de erro e comandos.

Quando há necessidade, a margem inferior avança para cima, adaptando-se ao tamanho de seu conteúdo.

A última linha inserida no programa chama-se linha corrente e é indicada pelo símbolo >.

O TK90X COMO CALCULADORA

3. O TK90X COMO CALCULADORA

Você pode utilizar seu computador como calculadora. Para tanto não há nenhuma complicação: basta que você empregue o chamado modo imediato da linguagem BASIC.

Neste capítulo mostraremos como fazer para que o TK9ØX efetue as operações algébricas através do modo imediato. Nos capítulos da unidade PROGRAMAÇÃO BASIC, você encontrará explicações acerca de operações matemáticas mais complexas, mas saiba que elas também funcionam no modo imediato.

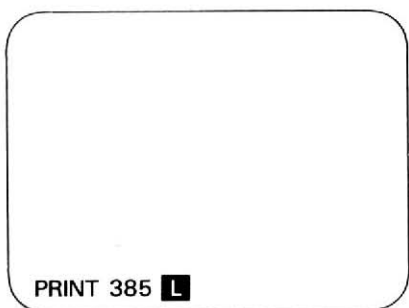
3.1. Modo Imediato

Modo imediato é o modo do processamento que o computador realiza quando você emprega a linguagem BASIC diretamente, ou seja, sem enumerar as linhas, sem estar escrevendo um programa. Neste modo, também denominado modo direto, o TK9ØX processa imediatamente as instruções, e a tela mostra o resultado da tarefa que você solicitou ao seu computador.

Faça dois pequenos exercícios:

a) Tecle o comando <PRINT> e o número 385.

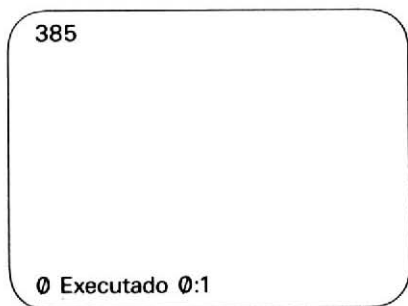
A tela mostrará:

```
PRINT 385 L
```

b) Pressione agora a tecla <ENTER>.

A tela mostrará:

```
385



Ø Executado Ø:1
```

3.2. Os Cálculos

A linguagem BASIC usa uma simbologia própria, para expressar as operações algébricas. Observe a tabela abaixo:

TIPO DE OPERAÇÃO	SÍMBOLO EMPREGADO
SOMA	+
SUBTRAÇÃO	−
MULTIPLICAÇÃO	*
DIVISÃO	/
POTENCIAÇÃO	↑

No modo imediato os cálculos algébricos são realizados através do comando PRINT mais o símbolo matemático correspondente, no BASIC, a cada tipo de operação. Somente a radiciação realiza-se de outra forma, como veremos mais adiante.

Experimente somar 8 e 2:

Para introduzir: Digite:

PRINT 8+2 <PRINT> <8> <SYMBOL SHIFT> e <K> <2> <ENTER>

O resultado (10) aparecerá no canto superior esquerdo da tela, a exemplo do que ocorreu quando você digitou apenas PRINT 385. Se você não desligou o computador desde então e se não deu nenhuma outra instrução, o número 10 aparecerá abaixo do 385, na segunda linha.

Tente obter uma divisão, por exemplo, 25/2 (verifique na tabela acima que símbolo o BASIC usa para representar a divisão).

Você deve ter digitado <PRINT> <25> </> <2> <ENTER> e obtido 12.5, abaixo do número 10.

Procure praticar as operações, variando números, símbolos algébricos e incrementando parcelas, multiplicadores etc.

Se você tentar uma divisão por zero, obterá uma mensagem de erro.

Nota: Mensagens de erro são tratadas especificamente no Apêndice B.

3.2.1. Potenciação

Para elevar um número a uma potência, você deverá digitar:

<PRINT> (base) <↑> (expoente), pressionando <ENTER> no final.

Exemplo:

Elevar 4 à quinta potência:

<PRINT> 4 <↑> <5> <ENTER>

Veremos o resultado na primeira linha disponível da parte superior da tela.

3.2.2. Radiciação

A radiciação é a operação matemática pela qual calculamos a potência fracionária de um número. Assim sendo, para calcular a raiz de um número, você deverá elevá-lo à potência de expoente igual a 1 dividido pelo radical.

Exemplo:

Calcularemos a raiz quinta de 9:

<PRINT> 9 <↑> (1/5) <ENTER>

Após o <ENTER> veremos a resposta: 1.5518456

3.2.3 Raiz Quadrada

Para extrair a raiz quadrada de um número, você tem duas opções:

a) Poderá teclar o comando <PRINT>, a função SQR (em azul acima da tecla <H>) e colocar o radicando após SQR. Finalize com <ENTER>.

Extrairemos a raiz quadrada de 16:

<PRINT> <CAPS SHIFT> e <SYMBOL SHIFT> <H> 16.

O resultado aparecerá na tela, abaixo da última linha utilizada.

b) Poderá proceder como no item 3.2.2.:

PRINT 16↑(1/2), que equivale a SQR 16.

3.3. Expressões Algébricas

Seu computador realiza as operações algébricas segundo uma ordem de precedência estabelecida:

1º) potenciação
2º) multiplicação/divisão
3º) soma/subtração

Se você quiser modificar esta ordenação, deverá empregar parênteses no lugar dos símbolos de associação.

Exemplos:

Calcularemos 25+10 – 5*2 – 75/3.

Obteremos 0 (zero) como resultado, pois o computador seguiu a ordenação estabelecida.

Calculando 25+(10 – 5)*2 – 75/3, o resultado será 10.

Para resolver [25+(10 – 5)]*2 – 75/3, substitua os colchetes por parênteses:

(25+(10 – 5))*2 – 75/3, e obterá 35.

Como no exemplo anterior, substitua as chaves e os colchetes da expressão 25+{10 – [5*2 – (75/3)]} por parênteses: 25+(10 – (5*2 – (75/3)))

Não esqueça do comando <PRINT>, no início, e de <ENTER> no final. O resultado será 50.

PROGRAMAÇÃO BASIC

5. PROGRAMAÇÃO BASIC

5.1. Conceitos

Até agora você teve de repetir várias vezes os comandos, para que o computador resolvesse seus exercícios. Os comandos ou instruções podem ser relacionados em linhas de programas, evitando-se, assim, que você tenha de repeti-los cada vez que desejar a execução de uma operação.

Ao conjunto de instruções com as quais se informam o computador, para que realize tarefas, dá-se o nome de "programa".

Os programas devem ser escritos em linguagem que o computador possa entender. O TK9ØX compreende a linguagem BASIC, portanto, você empregará esta linguagem para comunicar-se com seu computador.

Os Dados

Em linguagem BASIC distinguem-se dois principais tipos de dados: os numéricos e os alfanuméricos. Cada um deles recebe um tratamento diferente do computador.

Os dados numéricos são representados normalmente por algarismos (de Ø a 9).
Por exemplo: 23 e .Ø1234 são dados numéricos representados por algarismos arábicos.

Variáveis

As variáveis são armazenadores de valores aos quais são atribuídos determinados conteúdos. O conteúdo de uma variável pode ser de dois tipos: um valor numérico ou uma seqüência de caracteres entre aspas, denominada string. Assim sendo, o valor 5736 pode ser o conteúdo de uma variável numérica e "TK90X" o de uma variável string. Qualquer conteúdo disposto entre aspas é considerado um string, mesmo se utilizarmos uma seqüência de caracteres numéricos tal como "5.736". Neste caso estes algarismos não representam um valor, mas apenas uma cadeia de caracteres. A única exceção para o conteúdo de um string são as aspas. Um string não pode valer aspas. No entanto, os espaços em branco são considerados caracteres, logo, " " (um espaço entre aspas) pode ser atribuído como conteúdo de uma variável alfanumérica.

5736	TK90X
variável numérica	variável string

5.2. Comandos

PRINT

Você já teve oportunidade de experimentar o comando PRINT no modo imediato. Experimente-o agora num programa. Digite:

```
20 PRINT 2 + a <ENTER>
```

Desta forma, em vez de o comando ser executado, ele será armazenado numa linha de programa.

LET

Introduza mais uma linha, com numeração inferior à da tela:

```
10 LET a = 2
```

O comando LET atribui um valor a uma variável. À variável após o LET é atribuído o valor 2, expresso à direita do sinal de igualdade.

Quando você pressionar <ENTER>, a linha 10 passará da margem inferior para a posição correta que ocupa no programa, pois a ordenação crescente é obedecida, não importando em que momento a linha de programa foi introduzida. Sua tela deverá conter a listagem do programa:

```
10>LET a = 2
20  PRINT 2+a
```

Uma terceira linha será introduzida, desta vez com numeração intermediária:

```
15 LET b = 5
```

Nota: **Procure sempre numerar seus programas de forma que permita a introdução de novas linhas intermediárias. Aconselha-se a numeração de 10 em 10. As linhas podem usar a numeração de 1 a 9999.**

Pressione <ENTER> e a listagem será:

Com a introdução da linha 15, a linha 20 pode ser modificada, pois agora temos uma nova variável (b) e podemos introduzi-la no cálculo.

EDIT

Você poderia digitar novamente a linha 20, a fim de transformá-la, mas existe um meio mais prático, principalmente indicado no tratamento de linhas mais extensas, quando a nova digitação se tornaria enfadonha e demorada. Este meio mais prático de se modificar uma linha é editá-la.

A palavra-chave EDIT, sobre a tecla <1>, é acessada por <CAPS SHIFT> e <1> e edita, na margem inferior, a linha corrente. Esta é indicada por um cursor de linha-corrente, representado pelo sinal >. Note que este sinal está agora indicando a linha 15, a última a ser referenciada no programa.

Mova o cursor de linha-corrente, usando as teclas-setas <↑> <↓>, e pare-o na linha que você deseja editar. Neste momento você quer editar a linha 20, portanto, basta mover o cursor de linha-corrente uma posição para baixo. Estando o cursor posicionado, acesse <EDIT>, pressionando <CAPS SHIFT> e <1> simultaneamente. Você obtém então uma cópia da linha 20, na margem inferior da tela. Obtém ainda um cursor K, aguardando que você o desloque para a direita do caractere a ser modificado.

Para movimentar o cursor no sentido horizontal, utilize as teclas-setas <→> e <←>, que não alteram o conteúdo da linha, apenas fazendo passar o cursor sobre os caracteres.

Usando a tecla-seta à direita, (pressionando <CAPS SHIFT> e <8> simultaneamente), mova o cursor até uma posição além do número dois, antes do sinal +:

Nota: No modo de edição, o cursor salta automaticamente por sobre os caracteres de comando - experimente movê-lo para trás e para além do comando PRINT. Espaços adequados vão sendo criados, se você introduzir novos caracteres na linha que está sendo editada.

DELETE

Você tem com o cursor situado à direita do número 2, para substituir este número pela letra b . Se você simplesmente digitar b , este caractere será acrescentado à linha, mas o número 2 permanecerá. Há necessidade de eliminar-se este algarismo antes ou depois de se inserir o b . É mais prático eliminar antes de inserir, para se ter uma visão melhor do que está sendo editado e não ser preciso posicionar novamente o cursor.

Para eliminar o 2 , use o já conhecido comando <DELETE>. Pressionado <DELETE> apenas uma vez, você elimina o número e já tem o cursor posicionado para inserir a letra b. Faça-o e finalize a operação com <ENTER>.

Após este processo, seu computador mostrará:

Por enquanto o que foi feito não produziu resultados práticos. Temos um programa, mas até agora não obtivemos respostas para o cálculo que ele contém.

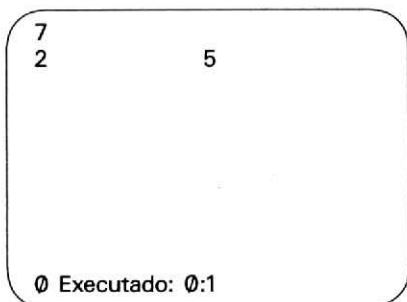
RUN

A execução de um programa requer que você dê uma ordem ao computador. Esta ordem é o comando <RUN>, seguido de <ENTER>. A palavra-chave RUN executa, faz "correr" ou "rodar" o programa. Rode-o e obterá na tela o resultado (7) do cálculo.

Nota: RUN pode ser usado seguido de um número de linha. Neste caso o programa será executado a partir daquela linha.

Faça o computador provar que ainda "lembra" as variáveis que foram criadas pelo seu programa. Digite em modo imediato:

PRINT a,b

No vídeo aparecerão, além do 7 já existente, os números 2 e 5 , na seguinte disposição:

Você já deve ter percebido que o número 5 apareceu na mesma linha que o 2 , mas algumas colunas adiante (coluna 16). Isto aconteceu porque você utilizou a vírgula entre as variáveis, no comando PRINT. Experimente outro PRINT, desta vez usando ponto e vírgula, assim:

<PRINT> a;b <ENTER>

O que você vê na tela não é o número 25, e sim os números 2 e 5 posicionados em colunas adjacentes. Se você tentar:

<PRINT> a <ENTER>

e em seguida:

<PRINT> b

as duas respostas usarão duas linhas distintas e subseqüentes. O mesmo acontece se você usar PRINT a 'b. O apóstrofo gera uma linha entre os elementos que ele separa.

CLS

Limpe a tela com <CLS> e <ENTER>. Feito isto, obtenha a listagem do programa, para introduzir mais alterações.

LIST

O comando <LIST> executa a operação de listagem.

Nota: De agora em diante não mais será necessário dizer-lhe que pressione <ENTER>, pois você já deve saber quando fazê-lo.

Você pode utilizar <LIST> para saltar o cursor de linha-corrente diretamente para a linha que quer editar.

Em listagens muito longas, seria bastante inconveniente mover o cursor linha após linha. É mais fácil comandar:

<LIST> (n.º de linha)

e depois <EDIT>

Vamos criar outras linhas em seu programa para exemplificar o que você acabou de aprender. Digite (não esqueça o <ENTER>):

```
30 <PRINT>
40 <PRINT>
50 <PRINT>
60 <PRINT>
```

Na tela você terá:

```
10  LET a=2
15  LET b=5
20  PRINT b+a
30  PRINT
40  PRINT
50  PRINT
60> PRINT
```

Repare na posição do cursor de linha-corrente. Suponha que você queira-o na linha 3Ø. Em vez de subi-lo com a seta, dê <LIST> 3Ø. Além de o cursor aparecer na posição desejada, acontecerá mais um fato: as linhas anteriores a 3Ø desaparecerão na nova listagem. Isto não tem importância, pois elas continuarão na memória do computador, e um próximo <LIST> as trará de volta. Por enquanto não necessitaremos delas.

A tela de seu vídeo comporta-se como uma janela através da qual você pode observar partes do programa armazenado na memória do TK9ØX.

O que seu vídeo apresenta agora é:

Está tudo pronto para usar <EDIT> e trazer uma réplica da linha 3Ø para a edição.

Aproveite a edição da linha 3Ø para acrescentar nela novos elementos.

Coloque o cursor **L** à direita de PRINT (use <→>). E digite "resultado". Não esqueça as aspas. Finalize a edição com <ENTER>. A listagem completa de seu programa reaparecerá na tela.

Vamos eliminar as linhas 4Ø, 5Ø e 6Ø, pois não fazem parte do objetivo do programa. Para eliminar uma linha, tecle o número dela seguido de <ENTER>. Cuidado com esta operação, ela é irreversível.

Restaram 4 linhas no programa (1Ø, 15, 2Ø e 3Ø). Rode-o. O resultado obtido não será muito lógico:

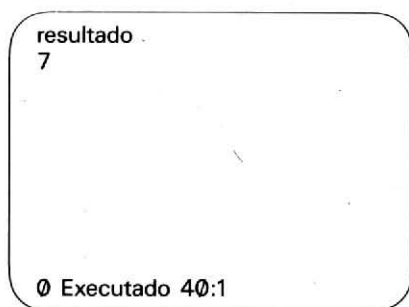
Apague tudo com <CLS> e liste o programa novamente.

Nota: **Para eliminar de vez os caracteres da margem inferior, sem apagar a tela toda e sem usar DELETE, pressione <EDIT>. Assim que a linha editada aparecer no fundo da tela, aperte <ENTER> e ela voltará ao programa, tendo esvaziado a faixa inferior do vídeo.**

Experimente editar a linha 2Ø e mudar sua numeração para 4Ø.

Se o cursor de linha-corrente não estiver aparecendo, liste a linha 2Ø ou acione a tecla de movimento <↑>. Isto o fará surgir.

Após corrigir a linha 2Ø, transformando seu número em 4Ø, encerre a edição e liste o programa. Você notará que a linha 2Ø permanece na listagem e que a linha 4Ø é uma cópia dela. Teremos que eliminar a 2Ø da maneira simples que você viu logo acima: digitando 2Ø <ENTER>.

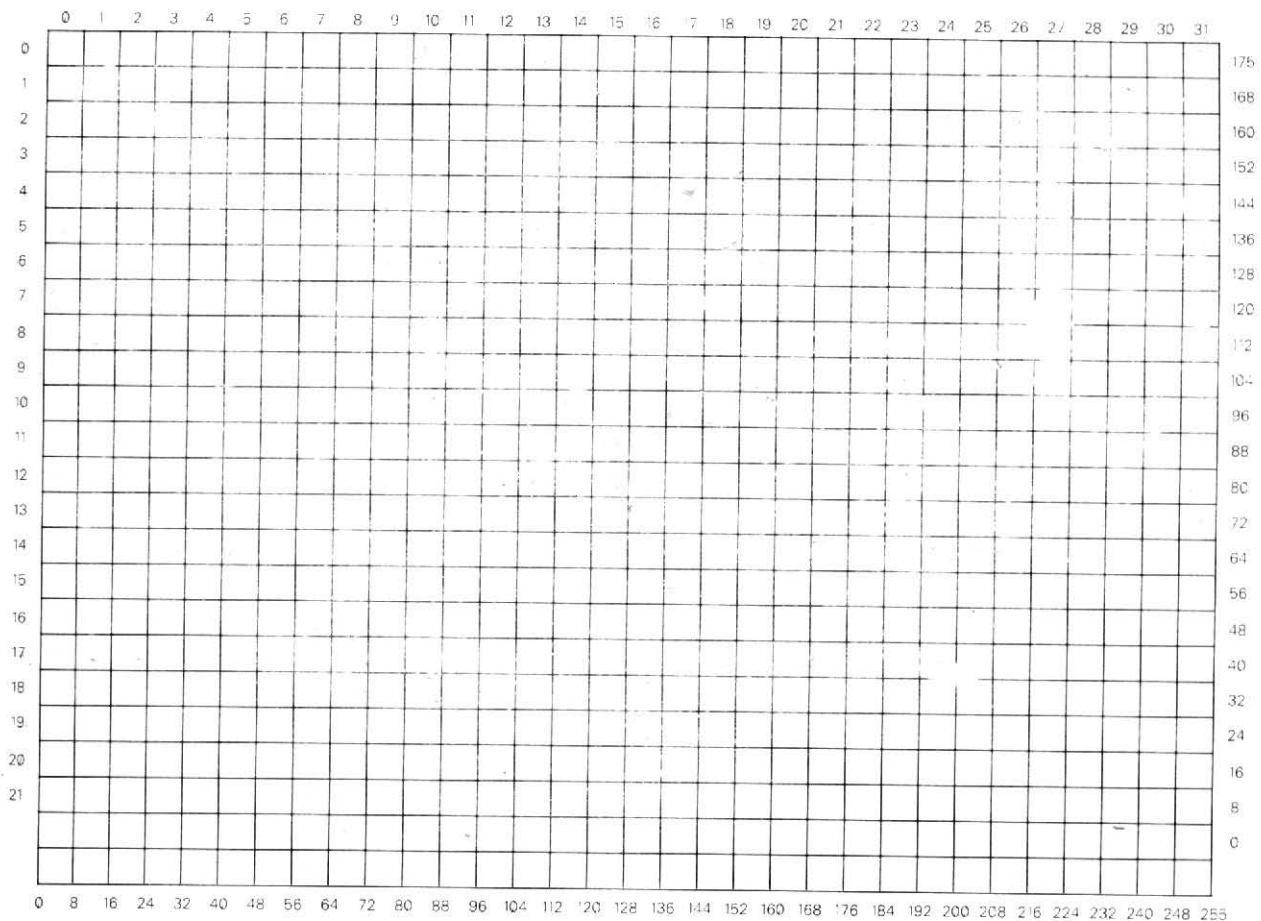
Agora está tudo pronto para tentar novamente. Pressione <RUN>. A saída do programa estará com um visual mais lógico:

O mesmo não se pode dizer quanto à estética. Para uma tela com 22 linhas e 32 colunas disponíveis, o aproveitamento do espaço deixa a desejar.

AT

Você pode controlar a disposição espacial aprendendo a usar a função AT, que vem precedida por PRINT. Sua forma é:

PRINT AT (linha), (coluna); (conteúdo)

Digite no modo imediato: <PRINT> <AT>11,1Ø;"(seu nome)" <ENTER>.

Seu nome deverá aparecer mais ou menos no centro da tela. Você pode ajustar a centralização, acertando os números de linha e coluna com o auxílio da matriz de tela de texto a seguir:

Quanto à apresentação visual do seu programa, vamos fazer alguma melhoria:

Limpe a tela e liste o programa.

Edite a linha 3Ø e modifique-a:

3Ø PRINT AT 9,12;"resultado"

Faça o mesmo com a 4Ø, variando linha e coluna:

4Ø PRINT AT 12,16;b+a

Rode o programa e você obterá:

O comando <PRINT> por si só encerra uma instrução de saltar uma linha. Teste isto introduzindo no programa novas linhas:

```
50 PRINT
60 PRINT "este e o PRINT da linha 60"
70 PRINT
80 PRINT "este e o PRINT da linha 80"
```

Nota: O PRINT que está entre aspas deverá ser digitado letra por letra, pois neste caso é apenas uma palavra e não um comando.

Ao rodar este programa você obterá:

TAB

Posiciona a coluna especificada para uso do PRINT. Seu maior valor numa linha corresponde a TAB 32 (última coluna). Caso este valor seja ultrapassado, os dados a serem impressos se deslocarão para a linha seguinte. Exemplo:

```
TAB 33 corresponde a TAB 1= 33 - 32
TAB 38 corresponde a TAB 6= 38 - 32
```

Uso prático:

```
PRINT TAB 6;"coluna 6";TAB 10;"coluna 10"
```

NEW

Modifique seu programa à vontade, aplicando todos os comandos que aprendeu até agora. Quando quiser apagar todo o programa da memória, utilize um comando que evita que você tenha de desligar o computador para apagar o conteúdo da RAM: o comando NEW, que atua até o RAMTOP.

NEW "esvazia" a memória do computador, destruindo todas as instruções e variáveis que foram acumuladas desde que você o ligou ou desde o último NEW.

É necessário sempre certificar-se de que você realmente deseja eliminar o conteúdo da memória RAM, pois é impossível recuperá-lo, a menos que tenha sido "salvo" numa fita cassete (como você verá em capítulo específico sobre o assunto).

Depois que você digitar <NEW>, o computador assumirá a atitude de início, como se você o tivesse ligado nesse instante.

Para fazer desaparecer a mensagem de apresentação, pressione <ENTER>. O cursor **K** surgirá logo a seguir.

REM

Se você desejar inserir na listagem de seu programa algum comentário ou mensagem, faça-o após a instrução REM, como no exemplo:

```
10 CLS
20 REM esta linha nao altera o programa
30 PRINT "meu nome e"
40 PRINT "(seu nome)": REM digite o nome
50 REM aqui termina o programa
60 PRINT "fim"
```

Esta instrução faz com que o computador ignore tudo o que você inseriu na linha, após o REM.

: (Dois pontos)

Você reparou nos dois pontos (:) da linha 40? Eles estão separando dois comandos do programa. Não houve necessidade de abrir-se uma outra linha de programa para a introdução do segundo comando. Observe que o cursor estava preparado para indentificar esta situação, tanto assim, que passou de **L** para **K** após a entrada dos dois pontos. Na verdade, você pode escrever vários comandos numa mesma linha separando-os apenas por dois pontos. Desse modo, você irá economizar espaço na memória de seu computador. Pode fazê-lo inclusive no modo imediato.

Rode o programa anterior. Depois introduza a linha abaixo:

```
55 PRINT: PRINT: PRINT
```

Faça o programa correr novamente para observar o efeito. Como era previsto, foram introduzidas três linhas em branco entre o seu nome e "fim".

Experimente usar os dois pontos em outros programas e habitue-se a introduzir REM em programas cuja listagem você quer preservar. Este recurso o ajudará muito na identificação do que o programa faz e como faz.

GOTO

É muito comum encontrarmos nos programas BASIC a instrução GOTO. Este comando faz com que o programa desvie para uma determinada linha.

Introduza uma linha em seu programa contendo esta instrução. Primeiramente, coloque-a com a numeração 25, como no exemplo:

```
10 CLS
20 REM esta linha nao altera o programa
25 GOTO 60
30 PRINT "meu nome e"
40 PRINT "(seu nome)": REM digite o seu nome
50 REM aqui termina o programa
55 PRINT: PRINT: PRINT
60 PRINT "fim"
```

Nota: Se você especificar uma linha inexistente após GOTO, o computador acessará a próxima linha existente. Se não houver nenhuma outra linha, o programa parará.

Execute o programa. Observe o resultado: ao atingir a linha 25, o programa salta diretamente para a linha 60, ignorando as linhas intermediárias.

Você pode colocar o GOTO em outras linhas do programa, para verificar como ele atua. Tente colocá-lo numa linha 70, eliminando a linha 25, da seguinte forma:

```
70 GOTO 30.
```

Rode o programa.

Você só sairá do "loop" quando pressionar <BREAK> ou responder negativamente ao "scroll?".

Você pode usar GOTO (n.º linha) no modo imediato, quando estiver com um programa na memória. É diferente de RUN (n.º de linha), pois este anula as variáveis, e GOTO não as altera.

INPUT

Abandone o programa e limpe a memória com <NEW>. A seguir digite:

```
10 PRINT "digite uma palavra"
20 INPUT a$
30 PRINT "digite um numero"
40 INPUT b
50 PRINT: PRINT "a$=";a$,"b=";b
60 PRINT: GOTO 10
```

Neste programa aparece um novo comando denominado INPUT. INPUT, assim como LET, permite a associação entre um nome de variável e seu valor. No entanto, INPUT detém o programa até que você introduza o dado que está sendo pedido.

A cada passagem pelo INPUT, você pode substituir o conteúdo da variável, seja ela numérica ou "string" (alfanumérica). Quando o computador estiver à espera do valor de uma variável numérica, o cursor ficará em **L**, e se colocará na margem inferior esquerda do vídeo.

Assim que você digitar o valor e <ENTER>, a variável assumirá o valor que você entrou.

Quando a variável for do tipo "string", o cursor no modo **L** estará localizado no mesmo lugar, mas, entre aspas. Isto é coerente com a necessidade de colocarem-se aspas nos valores "strings". O computador encarrega-se de colocá-las automaticamente no INPUT.

O INPUT pode incluir um comentário, mensagem ou qualquer "string", mas é necessário separar tal "string" da variável por um ponto e vírgula, na linha do programa. Este "string" aparecerá na margem inferior da tela, precedendo o cursor e terá por função informar o tipo de dado a ser introduzido naquele momento. Observe o seguinte exemplo:

```
<NEW>
10 CLS
20 INPUT "digite uma palavra"; a$
30 INPUT "digite um numero";b
40 PRINT "a$ =";a$,"b =";b
50 GOTO 20
```

Input Line

Uma variação bastante útil do comando INPUT é a instrução INPUT LINE.
Este comando permite a entrada de strings, que serão associados as respectivas variáveis. A diferença entre este INPUT e o que já havia sido apresentado, é que com LINE, o computador omite as aspas, que sempre acompanham o INPUT. Isto permite que você as introduza no string, tratando-as como caracteres quaisquer.

O formato desta instrução é
INPUT LINE variável string
Pode-se usar INPUT LINE também no modo imediato.

Nota: O INPUT LINE não aceita interrupção por meio do comando <STOP>.

STOP

Para deter um programa no momento do INPUT, use <STOP> (<SYMBOL SHIFT>+ <A>). Se o INPUT estiver à espera de um "string", elimine as aspas ou mova o cursor para a esquerda delas. Não o fazendo, o computador entenderá o STOP como o "string" que você está atribuindo à variável do programa e não o interromperá.

CONT

Se você pretende continuar a execução do programa após uma interrupção do tipo BREAK, STOP ou negativa a "scroll?", use <CONT> (continue). Esta instrução faz com que o computador retorne à execução do programa.

Quando o comando CONT é acionado após a mensagem "D BREAK-CONT repete", a retomada terá início no mesmo ponto em que parou, repetindo a última ação executada. Após a mensagem "L BREAK-CONT prossegue", a retomada terá início na primeira instrucão após a última ação executada.

A instrução CONT age de maneira peculiar quando você responde "n" à pergunta "scroll?" e o programa pára.

Experimente digitar 25 linhas de programa contendo apenas "REM", assim:

```
1 REM
2 REM
3 REM
4 REM
.
.
.
25 REM
<LIST> <ENTER>
```

Visto que na tela cabem apenas 22 linhas, aparecerá a mensagem "scroll?". Responda <n>. Outra mensagem aparecerá. Desta vez ela contém:

D BREAK CONT repete Ø :1

Responda com CONT <ENTER>

A margem inferior da tela desaparece. O computador está em "loop", porque CONT repete a primeira instrução da linha de comando: no caso, a instrução LIST. CONT será executada indefinidamente até que seja interrompida. Para fazê-lo, pressione <CAPS SHIFT> e <BREAK> simultaneamente. Você receberá a mensagem:

L BREAK - CONT prossegue Ø :1

Nada disso ocorre se colocarmos dois pontos (:) antes de LIST. Verifique::

: LIST <ENTER>

Quando aparecer "scroll?", digite <n>. Após a mensagem, entre <CONT>.

Desta vez surgirá:

Ø Executado Ø :1

porque a instrução CONT salta para a segunda instrução da linha, que é o fim. Se você usar:

: : LIST

CONT saltará para a terceira (e inexiste) instrução da linha, gerando a mensagem:

N Comando perdido Ø :2

Destas duas maneiras evita-se o "loop".

Antes de passar para o próximo capítulo, procure fazer alguns programas aplicando, se possível, todos os comandos que você aprendeu até aqui.

DECISÕES

6. DECISÕES

6.1. IF ... THEN ...

STOP

Dentre as características mais significativas do computador, destaca- se a sua capacidade de tomar decisões.

Até agora você viu o TK90X seguir precisamente as instruções que você programou, sem uma participação ativa na resolução de algum problema. No entanto, ele é capaz de ser bem mais útil, pois consegue resolver, além de problemas matemáticos, questões que envolvam lógica. Basta que você o programe para tal.

Um programa lógico poderia ter a seguinte estrutura:

```
10 Limpe a tela
20 Receba uma informação: Coloque na tela
30 Receba outra informação: Coloque na tela
40 Se a primeira for igual a segunda então vá para a linha 60
50 Escreva (1.ª informação) diferente (2.ª informação): Pare
60 Escreva (1.ª informação) = (2.ª informação)
70 Pare
```

em BASIC:

```
<NEW>
10 CLS
20 INPUT "entre informacao ";a$:PRINT AT 6,8;"1a. informacao ";a$
30 INPUT "entre informacao ";b$:PRINT AT 8,8;"2a. informacao ";b$
40 IF a$=b$ THEN GOTO 60
50 PRINT AT 12,10;a$;" <> ";b$:STOP
60 PRINT AT 10,8;a$;" = ";b$
70 STOP
```

Repare que, na linha 40, o programa força o computador a comparar os strings a$ e b$, e decidir:

Se a$ é igual a b$, o programa é desviado para a linha 60.

Se a$ é diferente (< >) de b$, o programa continua na mesma seqüência, isto é, na linha imediatamente posterior (50).

A função do STOP da linha 50 é bem clara: se ele não fosse colocado ali, o programa continuaria até o fim, desrespeitando as condições do IF.

É evidente que o seu TK90X pode trabalhar com programas bem mais difíceis e complexos, envolvendo strings e/ou valores a serem comparados, mas o princípio do método de comparação é similar ao que acabamos de ver.

O formato do IF é sempre IF (condição) THEN (instrução).

Tente substituir os strings por números ou expressões algébricas, sem esquecer-se de mudar as variáveis para numéricas.

Elabore um programa com maiores dificuldades e você comprovará a eficiência do seu computador como ferramenta de trabalho.

6.2. Comparando Dados

A seguir você tem uma tabela de operadores matemáticos e lógicos, para a comparação de dados:

SÍMBOLO BASIC	MATEM.	APLICAÇÃO
>	>	é maior que
<	<	é menor que
=	=	é igual a
<=	$\leqslant$	é menor ou igual a
>=	$\geqslant$	é maior ou igual a
<>	$\neq$	é diferente de (oposto a =)

OPERANDOS LÓGICOS	
AND (e)	relação AND outra relação
OR (ou)	relação OR outra relação
NOT (não)	relação NOT

Nota: < =, > = e < >, bem como AND, OR e NOT têm teclas próprias, não sendo necessário digitá-los caractere por caractere.

Estes operadores aplicam-se tanto para a manipulação de dados numéricos como de strings. Lembre-se, porém, de que não é possível comparar um string com um número e vice-versa.

6.2.1. Comparações Numéricas

Quando compara números, o computador segue os princípios da matemática. No programa a seguir, você verá como:

```
<NEW>
 10 INPUT "digite valor a ";a
 20 CLS
 30 INPUT "digite valor b ";b
 40 PRINT AT 6,5;"a= ";a
 50 PRINT AT 8,5;"b= ";b
 60 IF a>b THEN GOTO 100
 70 IF a=b THEN GOTO 120
 80 PRINT AT 10,5;a;" e menor do que ";b
 90 GOTO 10
100 PRINT AT 10,5;a;" e maior do que ";b
110 GOTO 10
120 PRINT AT 10,5;a;" e igual a ";b
130 GOTO 10
```

Nas comparações com os sinais <= e >=, basta que apenas uma condição seja satisfeita, para que a resposta seja considerada positiva, e seja cumprida a instrução do IF.

Por exemplo:

```
IF x <= Ø THEN PRINT "OK"
```

Para satisfazer à condição do IF, é indiferente que seja atribuído a x um valor menor que zero ou igual a zero.

As relações podem ser combinadas através do uso de operadores lógicos AND, OR e NOT, nos seguintes formatos:

uma relação AND outra relação

```
IF X < Ø AND A = 1ØØ THEN GOTO 12Ø
```

A instrução após o THEN somente é executada se ambas as condições são satisfeitas.

uma relação OR outra relação

```
IF y = Ø OR m = 1ØØ THEN STOP
```

A instrução STOP será obedecida desde que uma das condições seja satisfeita.

NOT relação

```
IF NOT P=Ø THEN PRINT P
```

O PRINT será executado desde que a relação seja falsa, isto é, desde que P seja diferente de zero.

6.2.2. Comparações Alfanuméricas

Podemos comparar não apenas números, mas também strings. Ao compararmos cadeias de caracteres devemos distinguir o significado que os sinais têm para strings do que têm para números. Os sinais "maior que" (>) e "menor que" (<) distinguem dois strings pela sua ordem, segundo o Código de Caracteres (Apêndice D), portanto:

```
"A" < "AA"          "BILHÃO" < "MILHÃO"
"AB" < "ABA"           "DEZ" > "CEM"
"AC" > "AB"
```

e assim por diante.

Os sinais <=, >=, <> e = atuam da mesma forma.

Exemplos, que poderão estar contidos em programas:

```
IF x$ = "SIM" THEN GOTO 3Ø
IF n$ <> "CARLOS" THEN PRINT N$
IF s$ > "FZZZZ" AND f$="FIM" THEN STOP
```

INSTRUÇÃO FOR... NEXT

7. INSTRUÇÃO FOR... NEXT

A instrução FOR ... NEXT compreende dois comandos: o comando FOR e o comando NEXT. Esta instrução faz com que sejam repetidas, um certo número de vezes, as instruções que se encontram entre os comandos FOR e NEXT.

Por exemplo, digite:

```
<NEW> <CAPS SHIFT> <2>
10 CLS
20 FOR I=1 TO 5
30 PRINT "TK90X"
40 NEXT I
<RUN>
```

Como você pode observar, o nome "TK90X" é apresentado cinco vezes na tela.

Vamos analisar o formato geral desta instrução:

```
FOR variável= valor inicial da variável TO valor final
.
.
.
linhas de instrução a serem repetidas
.
.
.
NEXT variável
```

Saiba como funciona o programa que você acabou de rodar:

O computador coloca o valor inicial na variável (a variável de uma instrução FOR ... NEXT tem de ser uma única letra).

O computador verifica se o valor inicial da variável "I" é menor que o valor final (5).

Se menor, faz com que o programa prossiga normalmente até encontrar o comando NEXT I (linha 40). Neste momento, o valor inicial da variável é acrescido de uma unidade.

O novo valor de I (agora I+1) é novamente comparado com o valor final da variável e, se ainda for menor, o programa salta para a linha seguinte a FOR.

O ciclo se repete até que o valor final de I iguale-se a 5. Ocorre então o último ciclo, pois quando passar por NEXT, I valerá 6 e o computador, ao comparar 6 com o valor final determinado para I, fará com que o "loop" seja abandonado, prosseguindo o programa na primeira instrução após o NEXT.

Nota: Sempre coloque após o NEXT a mesma variável que iniciou o ciclo (a que foi definida pelo FOR). Sempre use variáveis numéricas.

Pode-se inserir uma instrução FOR ... NEXT numa única linha, tal como:

```
20 FOR I=0 TO 100: PRINT I: NEXT I
```

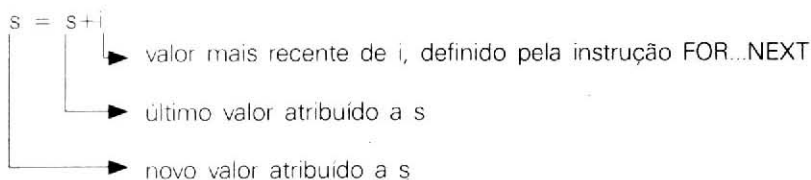
7.1. Uso de FOR ... NEXT como artifício matemático

O potencial desta instrução é muito grande. Neste item explicaremos a aplicação de FOR ... NEXT como artifício matemático dentro de um programa.

O programa exemplo que iremos descrever tem por função computar a soma dos inteiros positivos entre 1 e N. Digite:

```
<NEW>
 5 REM calculo da soma de inteiros positivos entre 1 e n
10 CLS
20 INPUT "n= "; n
30 LET s=0
40 FOR i= 1 TO n
50 LET s=s+i
60 NEXT i
70 PRINT "a soma e ";s
80 GOTO 20
```

A variável "s" (que no caso indica soma) é usada nas linhas 30, 50 e 70. Na linha 30, que é processada antes do comando FOR, o valor zero é atribuído a ela. A linha 50, entre os comandos FOR e NEXT, é executada para i=1, i=2, i=3 e assim por diante até i=n. O novo valor de "s" passa a ser:

```
s = s+i
|   |  └──► valor mais recente de i, definido pela instrução FOR...NEXT
|   └─────► último valor atribuído a s
└─────────► novo valor atribuído a s
```

Por exemplo, suponha que se atribua o valor 3 à variável "n"; então "s" terá os valores:

linha 30	s = 0
1.ª execução da linha 50	s = s+i = 0+1 = 1
2.ª execução da linha 50	s = s+i = 1+2 = 3
última execução da linha 50	s = s+i = 3+3 = 6

Execute o programa diversas vezes, variando sempre o valor de n.

7.2. STEP

Esta instrução faz com que se altere o "passo" da instrução FOR ... NEXT de 1 para o número que você desejar.

Recorde que cada vez que o programa passava em NEXT, o valor da variável indicada na ordem FOR era acrescido de uma unidade, até que se igualasse ao valor final. Pois bem, se junto a FOR houver a ordem STEP, a cada instrução NEXT, a variável passará a ser incrementada com o valor indicado após a ordem STEP. Este valor pode ser positivo ou negativo.

Observe o exemplo:

```
<NEW>
10 CLS
20 FOR j=0 TO 10 STEP 2
30 PRINT j
40 NEXT j
<RUN>
```

Quando o incremento é negativo, o programa continua o ciclo enquanto a variável determinada pelo FOR for maior ou igual ao limite.

Deve-se ter cuidado ao fazer dois ou mais "loops" FOR...NEXT ao mesmo tempo, ou seja, "loops aninhados". O programa abaixo mostra isto:

```
<NEW>
10 FOR m=0 TO 10
20 FOR n=0 TO m
30 PRINT m;":";n;":";
40 NEXT n
50 PRINT
60 NEXT m
```

O "loop" n está inteiramente dentro do "loop" m, portanto, estão enquadrados de maneira correta. É imperativo que os "loops" não se cruzem, pois se isto acontecer, o programa poderá não funcionar a contento.

Tente inverter as variáveis somente nas linhas 40 e 60, por exemplo. Você obterá um resultado bem diferente, pois a lógica do programa foi alterada.

Uma mensagem de erro do tipo "NEXT sem FOR" ou "Variável inexistente" será apresentada, caso você coloque um valor na variável inicial de FOR diferente do que colocou após o TO (isto não inclui o FOR ... NEXT com STEP negativo) ou, ainda, caso deixe de colocar o FOR, tendo introduzido o NEXT.

SUB-ROTINAS

8. SUB-ROTINAS

Por vezes, diferentes partes de um programa têm que executar a mesma tarefa, repetindo-se as instruções, desnecessariamente. Você pode evitar a repetição dessas linhas ao escrever o programa, tornando-as sub-rotinas e usando-as quando for necessário.

Para obter este efeito, use os comandos GOSUB e RETURN, no seguinte formato:

```
1Ø GOSUB n
2Ø
.
.
REM começo da sub-rotina
.
.
.
.
RETURN
```

Esta instrução significa: desvie para a linha n; execute todas as instruções até encontrar RETURN. Volte para a linha imediatamente seguinte a GOSUB.

Você entenderá mais claramente como funciona GOSUB e RETURN nos seguintes programas:

```
  <NEW>
  1Ø INPUT "entre um numero ";a
  2Ø IF a > 1Ø THEN GOSUB 1ØØ
  3Ø PRINT AT 1Ø,Ø;"sequencia normal do programa": PRINT
  4Ø PRINT "executando a linha 4Ø"
  5Ø PRINT: PRINT "linha 5Ø – fim – ":STOP
  6Ø REM comeco da sub-rotina
 1ØØ CLS
 11Ø PRINT AT 5,Ø;a;" > 1Ø"
 115 PRINT "houve desvio para linha 1ØØ"
 12Ø FOR c = 1 TO 5ØØ: NEXT c
 125 CLS
 13Ø RETURN
```

O exemplo a seguir demonstra como uma sub-rotina pode desviar para uma outra sub-rotina. Tente entender a lógica antes de rodar o programa.

```
 <NEW> - <CAPS SHIFT><2>
 15 PRINT AT 7,2;"O GOSUB DESVIA O PROGRAMA"
 20 PRINT TAB 2;"PARA UMA SUB-ROTINA A QUAL"
 25 GOSUB 500
 30 GOSUB 100
 35 PRINT TAB 2;"QUANDO ENCONTRA A INSTRUCAO"
 40 PRINT TAB 2;"RETURN."
 45 STOP
100 REM ******** SUB-ROTINA 100
105 PRINT TAB 2;"E PROCESSADA RETORNANDO"
110 PRINT TAB 2;"PARA A LINHA APOS O GOSUB"
115 GOSUB 500
120 RETURN
500 REM ******** SUB-ROTINA 500
505 FOR X=1 TO 700: NEXT X
510 RETURN
```

MODO TRACE: RECURSO DE DEPURAÇÃO

9. MODO TRACE: RECURSO DE DEPURAÇÃO

Utilizado como apoio na elaboração de programas, este comando é uma poderosa ferramenta para a depuração.

Quando se trabalha com programas complexos, com muitas condições e desvios para sub-rotinas, a possibilidade de erros de lógica torna-se maior, em decorrência das diversas opções da linguagem BASIC. O comando TRACE possibilita a monitorização das linhas executadas pelo programa, tornando-se assim possível a detecção de eventuais erros, e em geral, a verificação da execução do programa de acordo com as regras lógicas planejadas.

Formato: TRACE 1 (ativado) TRACE Ø (desativado)

Nota: O número da linha é impresso antes da execução. Em linhas com mais de um comando (separados por :) o número da linha será impresso para cada comando. TRACE sem o argumento equivale a TRACE Ø.

Digite o programa:

```
  <NEW>
 1Ø INPUT "tecle enter"; LINE e$:CLS: PRINT AT 5,8;"digite 1ØØ, 2ØØ ou 3ØØ
  2Ø INPUT a
  3Ø IF a = 1ØØ THEN GOTO 1ØØ
  4Ø IF a = 2ØØ THEN GOTO 2ØØ
  5Ø IF a = 3ØØ THEN GOTO 3ØØ
  6Ø STOP
 1ØØ PRINT AT 8,8;"esta e a linha 1ØØ"
 11Ø GOTO 1Ø
 2ØØ PRINT AT 8,8;"esta e a linha 2ØØ"
 21Ø GOTO 1Ø
 3ØØ PRINT AT 8,8;"esta e a linha 3ØØ"
 31Ø GOTO 1Ø
```

Antes de executar o programa, analise-o. Este é um programa propositadamente simples, para que você possa acompanhar seus desvios sem usar TRACE. Execute-o da maneira habitual. Feito isto, digite TRACE 1.

Rode o programa novamente e observe que agora os números das linhas executadas vão sendo mostrados após o sinal # . Desta forma, você pode saber exatamente para onde os desvios conduzem a execução das instruções.

Imagine se você tivesse que analisar um programa complicado e repleto de desvios condicionais e de subrotinas. Na certa não conseguiria acompanhar sua execução e localizar algum erro eventual. Com TRACE ativado, a tarefa seria bem mais cômoda.

Para melhor aproveitamento, você deve se colocar no lugar do computador, concluir qual deve ser a próxima linha a ser executada, e conferir com o que o comando TRACE está indicando.

TRACE pode ser ativado e desativado tanto no modo imediato, como em linha de programa.

TABELA DE DADOS

10. TABELA DE DADOS

10.1. READ ... DATA

Quando um programa requer a introdução de vários dados ou informações, o uso do comando INPUT torna-se cansativo e lento. Ele poderá ser substituído, com vantagem, pela instrução READ ... DATA.

As instruções READ e DATA são normalmente usadas em conjunto. Se um programa contiver uma ou mais instruções DATA, deverá ter pelo menos uma instrução READ.

Geralmente, por uma questão de organização, a instrução DATA é incluída no início ou no final do programa, porém, nada impede que ela seja colocada em qualquer outro ponto do programa. Veja os exemplos a seguir:

Aqui ela aparece no começo:

```
10 DATA 10,20,30,40,50
20 FOR v= 1 TO 5
30 READ a
40 PRINT "eu tenho agora ";a;" bananas"
50 NEXT v
```

Resultado:

```
eu tenho agora 10 bananas
eu tenho agora 20 bananas
eu tenho agora 30 bananas
eu tenho agora 40 bananas
eu tenho agora 50 bananas
```

Neste programa DATA foi introduzido no final:

```
10 FOR v= 1 TO 5
20 READ a$
30 PRINT "eu tenho "; a$;" bananas"
40 FOR i= 0 TO 300: NEXT i
50 NEXT v
60 DATA "poucas","muitas","quinze","laranjas e","uma dúzia de"
```

Resultado:

```
eu tenho muitas bananas
eu tenho quinze bananas
eu tenho laranjas e bananas
eu tenho uma duzia de bananas
```

10.2. RESTORE

Dissemos que o comando READ retira, enquanto lê, os dados da instrução DATA. Todos os dados terão sido usados ao término da leitura, portanto, o número de dados disponíveis deverá ser igual ao número de vezes que você indicou, no programa, que o computador efetuasse a leitura (e a retirada). Se a leitura prosseguir, ocorrerá uma mensagem de erro "E Fim de dados", a menos que você reconstitua a lista de dados, usando o comando RESTORE.

RESTORE (n.º linha)

Reabilita o conjunto de dados na memória a partir de um determinado ponto (volta o ponteiro a um ponto específico). Cada vez que ele é executado, os dados de DATA voltam a ser lidos desde o inicio, pela próxima instrução READ.

Rode este programa:

```
<NEW>
10 DATA "um TK90X"," manual ","gravador","televisao"
20 CLS
30 FOR v=1 TO 4
40 READ a$
50 PRINT AT 10,0;"eu tenho ";a$
60 FOR i=0 TO 350: NEXT i
70 NEXT v
80 GOTO 20
<RUN>
```

Acrescente a linha 75 ao programa. Digite:

```
75 RESTORE
<RUN>
```

A linha 75 faz com que os dados de DATA sejam reabilitados, permitindo nova leitura.

A instrução RESTORE pode ser usada para a reutilização de uma linha de dados específica, bastando, para isso, que se indique o número da linha.

Exemplo:

```
 <NEW>
  10 REM estacoes do ano
  20 DATA "primavera","verao","outono","inverno"
  30 REM pontos cardeais
  40 DATA "norte","sul","leste","oeste"
 100 REM leitura dos pontos cardeias
 110 RESTORE 40
 120 FOR i=1 TO 4
 130 READ a$
 140 PRINT "ponto cardeal ";i;": ";a$
 150 NEXT i
 200 REM leitura das estacoes do ano
 210 RESTORE 20
 220 FOR i=1 TO 4
 230 READ a$
 240 PRINT "estacao do ano ";i;": ";a$
 250 NEXT i
 300 STOP
```

20 definição dos dados referentes às estações do ano.

40 definição dos dados referentes aos pontos cardeais.

110 seleciona DATA da linha 40 (RESTORE número de linha).

120 a 150 rotina de leitura de pontos cardeais e impressão na tela.

210 seleciona DATA da linha 20 (RESTORE número de linha).

220 a 250 rotina de leitura de estações do ano e impressão na tela.

Obs.: Neste programa a utilização do RESTORE número de linha realça a versatilidade deste comando na manipulação de dados específicos, contidos em DATAS distintos.

Repare que neste caso inicialmente foi utilizado o comando RESTORE número de linha (110 RESTORE 40 e 210 RESTORE 20) antes da rotina de leitura, permitindo assim a seleção prévia dos dados a serem lidos.

Nas linhas 120 e 220, note que foram definidos loops de 4 ciclos, pois cada DATA contém 4 dados.

Ocorrerá a mensagem de erro E Fim de dados x:y caso o número de ciclos definidos nos loops FOR ... NEXT seja maior que a quantidade de dados contidos nos DATAS.

COMO DENOMINAR VARIÁVEIS

11. COMO DENOMINAR VARIÁVEIS

As variáveis, tanto numéricas como string, têm algumas normas de utilização:

As variáveis string devem ser representadas por uma única letra, seguida por um $.

As variáveis numéricas podem usar quaisquer letras ou dígitos (desde que o primeiro seja uma letra). Podem ser incluídos espaços, porém estes não serão considerados parte do nome. Pode-se usar tanto maiúsculas como minúsculas.

Exemplos de nomes válidos:

NUMÉRICAS	STRING
x	x$
VARIAVEL1	
TOTAL	Q$
t1	
mes12	L$

Exemplo de nomes incorretos:

1000 (é um número)
5 meses (começa com número)
A*N*O(* não é letra nem dígito)
Arco-Iris (– não é letra nem dígito)

Às variáveis numéricas podem ser atribuídos valores (dentro dos padrões explicados anteriormente) que podem ser representados inclusive por expressões ou outras variáveis já estipuladas dentro do programa.

Exemplo:

```
X = 2*a+tot1        mes1 = tot1/4*ano2        AA = soma+total
```

Os valores para variáveis string devem vir sempre entre aspas:

```
A$="banana"        n$="nome"        C$="contagem dos dividendos"
```

Quando se quer escrever um string contendo aspas, deve-se introduzir as aspas em dobro, ladeando a palavra em destaque.

```
R$ = "este e um string contendo" "aspas" "em dobro"
e$ = "este" "string" "e um exemplo"
```

MANIPULANDO STRINGS

12. MANIPULANDO STRINGS

Entende-se por string uma cadeia de caracteres consecutivos dispostos entre aspas, numa determinada seqüência. Se tomarmos apenas parte desta seqüência, teremos um substring. Veja os exemplos que seguem:

STRING	SUBSTRING
"1234567890"	"12345" "234" "567890"
"casa de cachorro"	"cas" "casa de" "de cachorro"

12.1. Determinando um "Corte"

Cada substring representa um corte do string original. É possível determinar um substring, a partir de um string maior, aplicando a função TO (não confundir com outras aplicações da função TO de outros comandos).

Esta função pode ser representada por uma notação bastante distinta em outras versões de BASIC (por exemplo, MID$, LEFT$ e RIGHT$). O comando TO pode ser representado no formato a seguir:

"seqüência de caracteres"(x TO y)

Exemplo:

STRING	CORTE	SUBSTRING
"sou o string original"	(3 TO 9)	"u o str"

No exemplo que vimos, o corte efetuado pode ser esquematizado assim:

```
so|u o s t r|ing original
  |         |
12|3456789  |
```

Nota: Observe que o espaço em branco é considerado um caractere.

Veja uma aplicação deste recurso:

```
10 LET a$ = "sou o string original"
20 PRINT a$
30 PRINT:PRINT a$(3 TO 9)
```

Rode e você obterá "u o str".

Se você varia o conteúdo dos parênteses na linha 30:

para a$ (TO 9) - "sou o str" para a$ (4 TO) - "o string original" para a$ (TO) - "sou o string original"

a$(TO) equivale a a$(1 TO final do string). O mesmo acontece com a$( ) ou mesmo a$.

Existe uma última forma, ainda, na qual o TO é omitido:

"sou o string original"(8)="t"   pois "t" ocupa a oitava posição na sequência.

Tente, por exemplo, substituir o índice y da linha 30 por 22:

```
30 PRINT:PRINT a$ (3 TO 22)
```

Quando o parâmetro após o TO for maior que o tamanho do string, aparecerá a mensagem de erro: 3 Erro de índice .

Modifique novamente a linha 30:

```
30 PRINT:PRINT a$(10 TO 5)
```

Neste exemplo, o resultado obtido é um string vazio, que pode ser representado por "".

Caso um dos índices (x ou y) seja negativo, ocorrerá outra mensagem de erro, dizendo: "Inteiro excede limite".

12.2. Mesclando Strings

Além de se poder "cortar um pedaço" de um string, é possível também "enxertar" um substring em outro ou num string original, criando os mais diversos arranjos:

Digite:

```
<NEW>
10 LET a$="eu sou um computador"
20 PRINT a$
30 LET a$ (5 TO 7)="***"
40 PRINT a$
<RUN>
```

Em seu vídeo deve aparecer:

```
eu sou um computador
eu s***um computador
```

Observe que os três asteriscos tomaram o lugar dos caracteres originais da posição 5 a 7 do string.

Tente aumentar o número de asteriscos entre as aspas, mas sem modificar os índices entre parênteses:

```
30 LET a$(5 TO 7)="******"
```

Parece que nada mudou:

```
eu sou  um computador
eu s***um computador
```

O que ocorre aqui é uma adaptação do substring "enxertado" ao espaço delimitado pelos índices x e y. Havia seis asteriscos para ocupar apenas três posições, então, três deles foram eliminados. É interessante saber que o computador eliminou-os a partir do extremo direito.

Coloque "123456" no lugar dos asteriscos. Verá que "456" não aparecerá no novo string. Quando o limite estipulado pelos índices x e y é maior que o substring que será enxertado, as posições restantes são automaticamente preenchidas por espaços em branco. Altere novamente a linha 30:

```
30 LET a$(3 TO 10)="12345"
```

Rode o programa e verá:

```
eu sou um computador
eu 12345 computador
```

Observe que os espaços em branco são caracteres significativos, pois tomaram três posições do string, sobrepondo-se ao que lá havia.

12.3. Somando Strings

É perfeitamente viável somarmos strings, desde que o termo "somar" seja interpretado como uma concatenação de caracteres de um determinado string (ou substring) com outro.

Experimente isto:

```
<NEW>
10 LET a$= "eu sou"
20 LET b$= "um computador"
30 LET c$= a$+b$
40 PRINT a$;b$
50 PRINT "a$+b$ = ";c$
<RUN>
```

Em seu vídeo você verá:

```
eu sou um computador
a$+b$ = eu sou um computador
```

Digite este programa e rode-o:

```
<NEW>
10 LET a$= "eu sou um computador"
20 LET b$=a$+a$(11 TO)
30 PRINT a$: PRINT
40 PRINT "a$+a$(11 TO) =": PRINT b$
```

A linha 30 imprime a$ e uma linha em branco.

A linha 40 imprime o que está entre aspas e, no comando seguinte, o valor de b$, que é a$ + "computador", já que cortamos este substring do string original a$, desde a décima primeira posição até o fim. Por este motivo, "computador" aparece duas vezes. Como não colocamos espaço nem no final do string a$ e nem b$ no início do substring, as palavras foram impressas sem intervalo. A ordem dos fatores neste caso muda o resultado.

Inverta a ordem dos comandos na linha 20 do programa e observe o resultado.

12.4. Manipulando o Comprimento do String - LEN

A instrução LEN faz com que o computador conte o número de caracteres de uma cadeia.

Pode-se usar LEN de duas maneiras:

```
LEN"(string)" ou LEN (nome de variável string)
```

Exemplo:

```
PRINT LEN "abelha"

LET a$="abelha"
PRINT LEN a$
```

Como você pode ver, nos dois formatos o resultado é o mesmo, ou seja, a apresentação do número 6 na tela, visto que "abelha" possui 6 caracteres.

Entre com o seguinte programa no seu computador:

```
<NEW>
10 REM contagem de caracteres
20 INPUT "qual a palavra?";x$
30 LET n= LEN x$
40 PRINT "o numero de letras e ";n
50 PRINT
60 GOTO 20
<RUN>
```

Repare que, neste exemplo, o número de caracteres do string foi atribuído ao parâmetro "nome de variável string" da instrução LEN.

Nota: Para que o programa pare, mova o cursor até a esquerda das aspas e entre STOP, conforme você aprendeu anteriormente.

FUNÇÕES

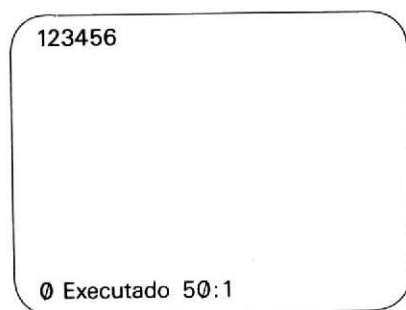
13. FUNÇÕES

Aqui são apresentadas mais algumas funções:

13.1. STR$ (número)

Como indica o cifrão $, esta função converte um número em string. Exemplo:

```
<NEW>
10 LET a=123
20 LET b=456
30 LET a$=STR$(a)
40 LET b$=STR$(b)
50 PRINT a$+b$
```

Após o comando RUN teremos:

Note que os dados 123 e 456 não são tratados como números (cuja soma seria 579), mas sim como strings. O sinal + fez com que se concatenassem.

13.2. VAL variável string ou VAL "string"

A instrução VAL executa a função inversa à da instrução STR$.

Observe o exemplo abaixo:

```
<NEW>
10 LET a$="123"
20 LET b$="456"
30 PRINT a$+b$
40 PRINT VAL a$+VAL b$
<RUN>
```

Após o RUN serão apresentados:

```
123456
579




Ø Executado 4Ø:1
```

Observe que, enquanto a soma dos strings "123" + "456" resulta em "123456", a soma do VAL deles resulta na soma verdadeira de seus valores numéricos.

13.3. VAL$ string

Apesar de muito semelhante à VAL, esta instrução não tem a mesma utilidade. O argumento de VAL$ é também um string ou variável string , no entanto, seu resultado permanece sendo um string, e não um valor numérico.

Exemplo:

```
<NEW>
1Ø LET a$="TK9ØX"
15 LET c$="a$"
2Ø PRINT "VAL$";"" ""c$"" "";" = ";VAL$ "c$"
25 PRINT
3Ø PRINT "VAL$ c$ = ";VAL$ c$
<RUN>
```

Acompanhe as instruções a seguir:

Primeiramente, introduza os valores das duas variáveis string, digitando em modo imediato:

```
<NEW>
LET A$= "" ""TK9ØX"" "" e        LET B$= "TK9ØX"
```

```
MEMÓRIA      "TK9ØX"        TK9ØX
            └───┬───┘     └───┬───┘
                A$            B$
```

Nota: **Lembre-se de que na memória do computador, a variável A$ contém "TK9ØX" com aspas por causa das aspas duplas e que B$ contém apenas TK9ØX, pois as aspas simples são simbólicas.**

Em seguida, solicite ao computador que retorne uma série de variações desses strings que estão na memória. Você obtém os resultados que colocamos ao lado de cada comando:

COMANDO	RESULTADO	COMENTÁRIO
PRINT A$ PRINT B$	"TK9ØX" TK9ØX	são os strings na forma original
PRINT VAL$ A$ PRINT VAL$ B$	TK9ØX C Erro de sintaxe 0:1	o VAL$ retira as aspas dos string originais. Como o string B$ ficou sem as aspas, tornou-se sintaticamente errado
PRINT "A $"	A$	este PRINT imprime o string que está entre aspas e que são os caracteres A e $
PRINT VAL$ "A$"	"TK9ØX"	novamente VAL$ retira as aspas, mas desta feita o PRINT se refere à variável A$, portanto a PRINT a$ que é = "TK9ØX"

Como você viu nos exemplos acima, VAL$ pode diversificar o uso de uma mesma variável alfanumérica. Isto pode ser útil para a seleção de elementos de matrizes multidimensionais ou até mesmo para o acesso das próprias matrizes. O uso de VAL$ tem efeitos mais palpáveis em programação avançada.

13.4. DEF FN

Esta instrução permite a definição de funções inexistentes no computador, criadas pelo próprio usuário. O formato da instrução é o seguinte:

DEF FN f(x) = fórmula

onde:

f = nome da função (qualquer letra do alfabeto)

x = variável da função (qualquer letra do alfabeto)

fórmula = qualquer função matemática utilizando a variável definida

A utilização desta instrução permite definir as mais diversas equações matemáticas, possibilitando a variação do argumento x num intervalo desejado.

Através da manipulação desta instrução, poderemos inclusive estabelecer o gráfico característico das equações utilizadas.

O exemplo a seguir aplica a fórmula da reta:

```
 5 DEF FN r(x)=2*x+b
1Ø PRINT "formula da reta"
2Ø PRINT AT 2,12;" 2*x+b"
25 INPUT "entre valor de x = ";x
3Ø INPUT "entre valor de b = ";b
4Ø PRINT "FN r(x)= ";FN r(x)
45 INPUT "prossegue (s/n)?"';"escolha a opcao e <enter>";p$
5Ø IF p$ <>"s" THEN STOP
55 CLS: GOTO 1Ø
```

Para a definição de funções utilizando strings, use o seguinte formato:

```
DEF FN f$(x$)= substring de x$
```

onde:

f$ = nome da função (qualquer letra seguida por cifrão)

x$ = nome da variável string (qualquer letra seguida de cifrão)

substring de x$ = qualquer função que, aplicada ao string, resulte num substring.

Exemplo:

```
DEF FN f$(x$)= x$ (6 TO)
```

Utilizando a fórmula acima, suponha que o string x$ seja:

"sou o computador TK90X"

Digite:

```
  <NEW>
  1 LET x$="sou o computador TK90X"
  5 DEF FN f$(x$) = x$(6 TO)
 10 PRINT FN f$(x$)
 <RUN>
```

O programa resulta no substring:

computador TK90X

No exemplo a seguir utiliza-se uma "fórmula" que gera sempre a metade de qualquer string que seja introduzido pelo INPUT:

```
  5 DEF FN f$(x$) = x$(LEN x$/2+1 TO)
 10 INPUT "entre um nome: ";x$
 20 PRINT FN f$(x$)
 25 INPUT "prossegue (s/n)";"escolha a opcao e <enter> ";p$
 30 IF P$ <>"s" THEN STOP
 35 CLS: GOTO 10
```

13.5. SGN — argumento

A função SGN indica o sinal do argumento:

```
PRINT SGN 123          PRINT SGN — 1.0034
```

O resultado será:

```
 1 se o sinal for positivo
-1 se o sinal for negativo
 0 se o argumento for zero
```

13.6. ABS argumento

É outra função que é usada somente para argumentos numéricos. Converte o argumento num número positivo, desconsiderando o sinal que o precede:

```
PRINT ABS - 30.5
      ou
PRINT ABS 30.5
```

Terão como resultado um único número positivo: 30.5.

13.7. INT argumento

Tem por finalidade obter apenas a parte inteira de um número (positivo ou negativo), ignorando a parte fracionária:

```
PRINT INT 30.9 resultará em 30,
    no entanto,
PRINT INT – 30.9 terá – 31 como resultado.
```

Você deve ter percebido que INT sempre arredonda para menos, ou seja, fornece o próximo número inteiro menor que o argumento.

FUNÇÕES MATEMÁTICAS

14. FUNÇÕES MATEMÁTICAS

14.1. EXP

O uso da matemática em áreas como a Engenharia e a Física requer a utilização de expoentes naturais que têm o número **e** (2,7182818) como base.

Se **e** elevado a x é igual a y, podemos usar a seguinte fórmula para o cálculo de y:

```
y= e↑x
```

então podemos expressar:

```
y= EXP(x)
```

14.2. LN

Logaritmo natural (base **e** = 2.71828182...). A função inversa da exponencial é a função logarítmica:

Exemplo:

```
PRINT LN 2/LN 1Ø

Resposta: Ø.3Ø1Ø3
```

Se você quiser calcular o logaritmo de um número em outra base, lembre a fórmula:

$\log_b a$= LNa/LNb

No exemplo anterior calculamos log 2 (base 1Ø).

14.3. PI

A função PI do TK9ØX (obtida através da tecla <M>, no modo extendido) é de grande valia quando se trabalha com cálculos que envolvem este número. O exemplo mais comum é o do cálculo da circunferência ou do perímetro de um círculo. Para efetuar este cálculo no seu micro, introduza:

```
LET P=d*PI
```

onde:

P = o perímetro que você procura
d = o diâmetro que você vai informar ao computador
PI = a função que representa o número 3.1415927

14.4. FUNÇÕES TRIGONOMÉTRICAS

SIN	COS	TAN	ASN	ACS	ATN
SENO	CO-SENO	TANGENTE	ARCO-SENO	ARCO-CO-SENO	ARCO-TANGENTE

Obs.: As funções trigonométricas trabalham em radianos, não em graus. Para efetuar conversões, use o seguinte esquema:

de radianos para graus	r/PI * 18Ø
de graus para radianos	g/18Ø * PI

Exemplo:

```
<NEW>
1Ø LET r= 36Ø/6.28
2Ø FOR i= Ø TO 36Ø STEP 3Ø
3Ø LET y= SIN (i/r)
35 LET x= COS (i/r)
4Ø LET z= TAN (i/r)
45 PRINT' ' "SIN ";i;" = ";y
5Ø PRINT"COS ";i;" = ";x
6Ø PRINT"TAN ";i;" = ";z
65 NEXT i
```

NÚMEROS RANDÔMICOS

15

15. NÚMEROS RANDÔMICOS

O TK90X trabalha com duas instruções distintas, no tratamento de números randômicos: RND e RAND.

15.1. RND

É similar a outras funções, dispensando, no entanto, qualquer argumento.
Use <PRINT> + <SYMBOL SHIFT> + <T> para acessá-la. RND gera um número aleatório entre 0 e 1, cada vez que você usá-la. Você pode obter inclusive o número 0, mas nunca o número 1.

Tente executar:

```
10 PRINT RND,
20 GOTO 10
```

Você obtém vários resultados. Note que é impossível determinar um padrão comum entre esses números. Podemos dizer que RND simula um número randômico, pois, na verdade, gera um pseudo-randômico, visto que segue uma seqüência fixa de 65.536 números.

Apesar de RND gerar números apenas na faixa entre 0 e 1, você pode ampliar esta faixa para 0 a 5, por exemplo, executando 5*RND.

Para obter números inteiros, use INT, sem esquecer que INT sempre arredonda para menos:

```
10 PRINT INT (6*RND),
20 GOTO 10
```

Com o programa acima você obtém um "loop" que gera números aleatórios entre 0 e 5.

15.2. RAND

Embora parecido com RND, RAND (de randomize) tem outra aplicação. É, na verdade, um comando e não uma função.

Você pode usar RAND para inicializar RND em diversos lugares da seqüência, digitando RAND e um número entre 1 e 65535, e <ENTER>.

No exemplo:

```
<NEW>
10 RAND 1
20 FOR t=1 TO 10
30 PRINT RND,
40 NEXT t
50 PRINT
60 GOTO 10
```

A cada passagem pela linha 10, a seqüência de RND recomeça com 0.0022735596.

Utilizando RAND sem argumento (ou com argumento zero, RAND Ø), o efeito será diferente do RAND da linha 1Ø, com argumento 1. Verifique isto no programa:

```
<NEW>
1Ø RAND
2Ø PRINT RND
3Ø GOTO 1Ø
```

Aqui não se obtém uma seqüência aleatória, pois RAND está usando a contagem desde que o computador foi ligado até a execução do programa (e durante este). Acontece que cada execução seguinte à primeira tomará da seqüência números muito próximos, perdendo, assim, o caráter aleatório.

Para que a escolha do número se torne mais aleatória, modifique a linha 3Ø para:

```
3Ø GOTO 2Ø
```

O uso desta função associada a comandos do tipo IF...THEN permite gerar programas com comportamento aleatório, como o requerido em jogos, simulações estatísticas e outras aplicações.

MATRIZES

16. MATRIZES

MATRIZ é um conjunto ou tabela de variáveis com um mesmo nome.

Cada elemento da matriz é identificado por um único índice numérico, que aparece entre parênteses, após o nome da matriz.

Exemplo:

a(12) = 245

a = nome da matriz
12 = índice que identifica um elemento da matriz **a**
245 = o elemento da matriz

Quando os elementos de uma matriz são dados alfanuméricos, o nome da matriz deverá ser seguido de cifrão ($).

Exemplo:

b$(5) = "L"
b$ = nome da matriz alfanumérica
5 = índice
"L" = elemento

16.1. DIM

Antes de podermos usar qualquer matriz, devemos reservar espaço para ela na memória do computador. Para tanto, utilizamos a instrução DIM (dimensionamento).

Suponha que você queira montar uma matriz contendo um conjunto de dez variáveis. Empregue DIM, da seguinte forma:

DIM a(10) - para variáveis numéricas
ou
DIM a$(10) - para variáveis alfanuméricas

Usaremos, como exemplo, uma matriz numérica **a** contendo dez elementos (de **a**(1) a **a**(10)). O programa abaixo, além de dimensionar a matriz em questão, atribui a cada elemento desta um valor de x. A operação é coordenada por um "loop" FOR...NEXT de dez ciclos.

```
<NEW>
10 DIM a(10)
20 FOR x=1 TO 10
30 LET a(x)=x
40 NEXT x
```

Para certificar-se de que sua matriz contém os elementos de 1 a 10, você poderá solicitar qualquer um deles ou todos:

```
PRINT a(5) - deverá resultar 5
```

ou

```
50 FOR y= 1 TO 10
60 PRINT "a(";y;")=";a(y),
70 NEXT y
```

Nota: Não use zero como valor inicial de um FOR...NEXT, se está trabalhando com matrizes, pois o zero não serve como índice de elementos.

16.2. Matrizes Multidimensionais

Em matrizes multidimensionais, pode-se associar vários valores a uma mesma variável.

O programa a seguir mostra como usar uma matriz bidimensional numérica e fornece seu resultado:

```
10 DIM a(5,2)
20 FOR c=1 TO 5
30 FOR f=1 TO 2
35 INPUT "entre um numero ";a(c,f)
40 NEXT f: PRINT AT 2,2;"insercao ";c
45 NEXT c
50 CLS
55 FOR v=1 TO 5
56 PRINT "conjunto ";v
60 FOR g=1 TO 2
65 PRINT "a("; v; ",";g;")";" = ";a(v,g)
70 NEXT g: PRINT: NEXT v
```

As linhas 10 a 45 correspondem à inserção de dados numéricos numa matriz bidimensional.

As linhas 55 a 70 mostram como os dados são identificados.

No exemplo temos a matriz representada em cinco linhas por duas colunas:

	Coluna 1	Coluna 2	
			linha 1
a =			
		100	
			linha 5

Para certificar-se de que sua matriz contém os elementos de 1 a 10, você poderá solicitar qualquer um deles ou todos:

```
PRINT a(5) - deverá resultar 5
```

ou

```
50 FOR y= 1 TO 10
60 PRINT "a(";y;")=";a(y),
70 NEXT y
```

Nota: Não use zero como valor inicial de um FOR...NEXT, se está trabalhando com matrizes, pois o zero não serve como índice de elementos.

16.2. Matrizes Multidimensionais

Em matrizes multidimensionais, pode-se associar vários valores a uma mesma variável.

O programa a seguir mostra como usar uma matriz bidimensional numérica e fornece seu resultado:

```
10 DIM a(5,2)
20 FOR c=1 TO 5
30 FOR f=1 TO 2
35 INPUT "entre um numero ";a(c,f)
40 NEXT f: PRINT AT 2,2;"insercao ";c
45 NEXT c
50 CLS
55 FOR v=1 TO 5
56 PRINT "conjunto ";v
60 FOR g=1 TO 2
65 PRINT "a("; v; ",";g;")";" = ";a(v,g)
70 NEXT g: PRINT: NEXT v
```

As linhas 10 a 45 correspondem à inserção de dados numéricos numa matriz bidimensional.

As linhas 55 a 70 mostram como os dados são identificados.

No exemplo temos a matriz representada em cinco linhas por duas colunas:

	Coluna 1	Coluna 2	
			linha 1
a =			
		100	
			linha 5

Os espaços que o string não ocupa no seu campo são preenchidos com brancos.

O string que ultrapassa os limites estabelecidos é truncado - cortado pelo extremo direito.

z$(2,5 TO 8) = z$(2) (5 TO 8) = GRAF. Este processo chama-se "slicing" e resulta, como se vê, no corte de um elemento da matriz.

16.3. Matrizes Tridimensionais

```
<NEW>
10 DIM a$(5,2,20)
20 FOR c=1 TO 5
30 FOR f=1 TO 2
35 PRINT AT 2,2;"insercao "; c
40 INPUT "entre um nome "; a$(c,f)
45 NEXT f: NEXT c
50 CLS
55 FOR v=1 TO 5
60 PRINT
65 PRINT "conjunto ";v
70 FOR g= 1 TO 2
75 PRINT "a$(";v;",";g;")";"= ";a$(v,g)
80 NEXT g: NEXT v
```

Na linha 10 encontra-se o dimensionamento da matriz:

```
DIM a$(5,2,20)
```

Onde:

o primeiro índice (5) determina que a matriz está disposta em cinco linhas

o segundo índice (2) informa que existem duas colunas

o último índice (20) estabelece a quantidade máxima de caracteres que podem ser introduzidos em cada elemento da matriz

Nota: Caso seja dimensionada uma matriz alfanumérica com um nome de variável já utilizado - seja para uma variável simples, seja para outra matriz - a última matriz prevalecerá, sendo as anteriores destruídas.

CORES 17

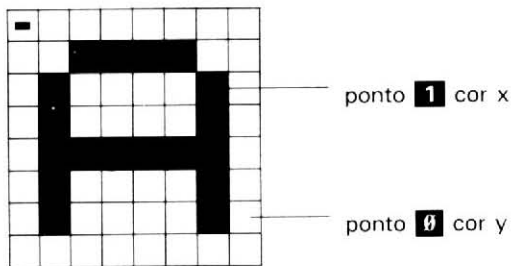
17. CORES

O TK9ØX pode produzir oito cores distintas e dois níveis de brilho. Caso o televisor seja em preto e branco, estas cores servirão para gerar tonalidades que variarão do cinza claro ao preto.

Abaixo, a lista das oito cores disponíveis:

Ø BLACK	(preto)
1 BLUE	(azul)
2 RED	(vermelho)
3 MAGENTA	(magenta)
4 GREEN	(verde)
5 CYAN	(ciano)
6 YELLOW	(amarelo)
7 WHITE	(branco)

Estas cores seguem, respectivamente, as teclas numéricas.

A imagem ativa é formada por uma área dividida em 24 linhas e 32 colunas, criando assim 768 posições. Cada uma destas posições é impressa no vídeo como um quadrado de 8 × 8 pontos. Isto permite que em cada posição possa ser representado qualquer caractere.

17.1. INK e PAPER

O ponto 1 representa um ponto aceso na matriz. Sua cor é definida de duas maneiras:

ao ligar-se o computador

imediatamente após o comando INK

Na primeira ocorrência, qualquer ponto aceso aparecerá na tela, na cor padrão (preto-Ø). No segundo caso, o ponto aparecerá na cor definida pela última instrução INK.

O ponto Ø representa um ponto apagado na matriz. Sua cor também é definida de duas maneiras:

ao ligar-se o computador, assumindo a cor branca (7)

imediatamente após o comando PAPER, com a cor definida pelo usuário

Exemplo:

```
INK 2
PAPER 6
PRINT "TK9ØX"
```

No exemplo acima, todos os pontos 1 assumem a cor 2 (vermelho), e os pontos Ø, a cor 6 (amarelo).

Tanto INK como PAPER podem ser definidos pelo usuário, na faixa de Ø a 7. Um valor maior atribuído, ocasionará uma mensagem de erro.

Existem outros meios de se usar as instruções INK e PAPER.

Digite:

```
<NEW>
1Ø PRINT PAPER 6;''amarelo'';: PRINT '' sol''
15 PRINT
2Ø PRINT INK 2;''vermelho'';: PRINT '' fogo''
<RUN>
```

Pode-se observar no resultado deste programa que tanto INK como PAPER voltam ao seu valor anterior no comando PRINT, após os dois pontos.

Este método pode também ser usado no comando INPUT, gerando o mesmo resultado que no PRINT:

```
<NEW>
1Ø INPUT INK 3; ''que cor e esta'';c$
15 IF c$< > ''magenta'' THEN GOTO 1Ø
2Ø INPUT PAPER 6; ''gostou da cor?'';g$
```

17.2. BRIGHT

Aumenta a intensidade da cor:

BRIGHT Ø = desligado

BRIGHT 1 = ligado - aumento da intensidade

Digite e execute o seguinte programa:

```
<NEW>
1Ø PRINT  PAPER 5;AT 2,1Ø;''programa teste''
2Ø PRINT AT 4,Ø;''observe o fundo da tela''
3Ø PRINT ''apos o comando BRIGHT''
35 PRINT INK 3;''veja na listagem a seguir:''
4Ø FOR x=1 TO 1ØØØ:NEXT x
5Ø BRIGHT 1
6Ø PRINT:LIST
65 BRIGHT Ø
```

Você deve notar como a cor de fundo da listagem aumenta seu brilho.

17.3. FLASH

Artifício usado para realçar um caractere, conjunto de caracteres ou até mesmo a tela inteira, onde:

FLASH Ø = desativado

FLASH 1 = ativado

Exemplo:

```
<NEW>
10 FLASH 1
20 PRINT PAPER 6;AT 10,10;"pisca";:PRINT PAPER 7;"-"; PAPER 5;"pisca"
30 FLASH 0
```

Caso se omita a linha 3Ø deste programa, FLASH não será desativado e a própria listagem sofrerá o efeito dele, quando LIST for efetuado. Mesmo que se desligue FLASH (FLASH Ø), o conteúdo da impressão continuará a piscar. Deve-se limpar a tela para interromper o efeito de FLASH, neste caso.

17.4. BORDER

Esta instrução tem como finalidade modificar as cores da borda da tela. BORDER Ø a 7 é o seu limite.

Exemplo:

```
<NEW>
10 FOR X=0 TO 7
20 BORDER x
25 PRINT INK 7;PAPER x;AT 10,2;"border cor ";x
26 IF x=7 THEN PRINT INK0; PAPER x; AT 10,2;"border cor ";x
30 FOR t=0 TO 500: NEXT t
35 NEXT x
40 GOTO 10
```

17.5. INVERSE

Instrução que possibilita a inversão das cores dos caracteres para a cor de fundo (PAPER), onde:

INVERSE Ø = desativado

INVERSE 1 = ativado

Quando se ativa o modo inverso, todos os caracteres contidos na tela sofrem alteração de cor, até mesmo a listagem do programa a seguir:

```
<NEW>
10 INK 3
20 INVERSE 1
25 PRINT AT 10,2;"caractere branco/fundo magenta"
30 FOR t=0 TO 1000: NEXT t
40 INVERSE 0
50 PRINT AT 10,2;"caractere magenta/fundo branco "
55 FOR t=0 TO 1000:NEXT t
60 GOTO 20
```

Note que, ao executar uma listagem, as linhas de programa também são invertidas.

Para voltar ao modo normal, entre:

```
INK 0
INVERSE 0
LIST
```

Utilizando os caracteres de controle também se obtêm as modificações necessárias.

Segue abaixo uma tabela de caracteres de controle:

CHR$	INSTRUÇÃO
16	INK
17	PAPER
18	FLASH
19	BRIGHT
20	INVERSE

Nota: As explicações referentes a CHR$ encontram-se no próximo capítulo.

Cada um dos caracteres de controle vistos anteriormente é seguido por outro caractere, que indica uma cor com o respectivo código:

```
<NEW>
10 PRINT CHR$ 17+CHR$ 4;"cor verde"
<RUN>
```

Outra aplicação dos caracteres de controle é o seu uso em linhas de instruções, onde se pode dar ênfase a uma linha ou sub-rotina do programa.

Siga as instruções:

Defina uma linha de programa - 10, por exemplo.
Vamos fazer um "loop" de 10 ciclos usando FOR ... NEXT, sendo esta linha (10) listada na cor verde. Então:

Digite 10.

Pressione <CAPS SHIFT> e <SYMBOL SHIFT> simultaneamente, para acionar o modo expandido, cursor **E**.

Defina a cor verde, pressionando <CAPS SHIFT> e <4>.

Agora basta digitar a linha:

FOR x= 0 TO 10

A linha deverá estar verde. Para voltar à cor anterior, faça:

<CAPS SHIFT> e <SYMBOL SHIFT> (cursor E)

Defina a cor anterior <CAPS SHIFT> e <n.º> <ENTER>

Este é o método para colorir as linhas de programas. Suponhamos que você queira enfatizar uma mensagem, fazer com que esta fique piscando numa linha de programa ou após sua impressão:

Faça, por exemplo, a linha 20, na cor ciano (5), usando a mesma regra anterior:

Digite: 20 PRINT"

Após ter digitado a linha 20 (20 PRINT"), acione o modo expandido (cursor **E**).

Pressione <CAPS SHIFT> <9> (liga o pisca).

Entre em modo expandido novamente.

Defina a cor amarela: <CAPS SHIFT> <6>.

Escreva a mensagem: FELICIDADE.

Após ter escrito a mensagem, deve-se desligar o pisca caso contrário, as linhas posteriores receberão o mesmo efeito. Para fazê-lo:

Entre no modo espandido (cursor **E**).

Acione <CAPS SHIFT> <8>.

Não se esqueça de mudar a cor para a desejada senão, a próxima linha receberá a última cor definida. Para fazê-lo:

Entre no modo expandido <CAPS SHIFT><SYMBOL SHIFT>.

Defina a cor anterior <CAPS SHIFT><n.º>

Feche as aspas.

<ENTER>

Agora, na linha 30, feche o "loop" (NEXT X) com a cor magenta (3), usando o mesmo método inicial (linha 10).

Rode o programa: <RUN>.

Obs.: Os números das linhas devem ser digitados antes de qualquer processo de coloração.

17.6. OVER

Utilizado para sobrepor literalmente strings numéricos, alfanuméricos ou caracteres isolados, o comando OVER (1 ativa, 0 desativa) é um interessante artifício na programação. Digite e execute o programa a seguir:

```
<NEW>
10 PRINT AT 20,1;"pressione ENTER"
15 LET a$=" OVER 0 teste"
20 LET b$=" OVER 1 teste"
25 LET c$="<        =  >        "
30 PRINT INK 1;AT 10,8;OVER 0;c$
35 INPUT e$
40 PRINT INK 2;AT 10,8;OVER 0;a$
45 INPUT e$
50 PRINT INK 3;AT 10,8;OVER 0;b$
55 INPUT e$
60 PRINT INK 1;AT 10,8;OVER 1;c$
65 INPUT e$:GOTO 10
```

15 a 25 - Definem os strings usados no exemplo.

30 - Imprime no centro da tela (AT 10,8), o conteúdo de c$, na cor amarela, com OVER 0 (desativado).

40 - Imprime o conteúdo de a$ na mesma posição de c$, na cor verde e com OVER 0 (desativado). Repare que a$ "ocupou" a posição de c$, eliminando-o.

50 - Imprime o conteúdo de b$, também ocupando o lugar da última mensagem.

60 - Imprime o conteúdo de c$ na mesma posição de b$, mas com OVER 1 (ativado). Note que quando utilizamos OVER 1, c$ sobrepõe-se a b$, porém, conserva seu conteúdo na tela, originando uma fusão dos dois strings.

Nota: O INPUT e$ das linhas 35, 45, 55, 65 funciona apenas como uma espécie de temporizador, pois o computador aguarda sua autorização para prosseguir.

17.7. ATTR

Fornece um número que retrata a situação de uma posição da tela - ATTR (linha, coluna). Originado pela somatória de quatro números que correspondem a:

128, se FLASH 1, ou Ø se FLASH Ø +

64, se BRIGHT 1, ou Ø se BRIGHT Ø +

8 vezes o número da cor definida com PAPER +

Ø número da cor definida com INK

Exemplo:

```
PRINT AT Ø,Ø;FLASH 1;BRIGHT 1;PAPER 6;INK 1;"$";ATTR(Ø,Ø)
```

O resultado obtido é 241 e refere-se a:

```
FLASH 1;BRIGHT 1;PAPER 6;INK 1
   V      V        V      V
  128 +   64    + (8*6)  + 1  = 241
```

17.8. INV. VIDEO E TRUE VIDEO

Estas funções inserem códigos de controle em linhas de programa para produzir cores invertidas ou normais. INV. VIDEO gera inversão de cores e TRUE VIDEO faz com que a inversão seja cancelada.

O CONJUNTO DE CARACTERES

18. O CONJUNTO DE CARACTERES

O conjunto de caracteres utilizado pelo TK90X é, na sua grande maioria, do mesmo tipo que o de outros computadores: o do código ASCII.

Além dos caracteres considerados "padrão", tais como os alfabéticos (maiúsculos e minúsculos), os numéricos (de 0 a 9) e os simbólicos comuns ($,< >,% etc.), o conjunto de caracteres do TK90X inclui as palavras-chaves (PRINT, LIST, LOAD etc.). Mas isto não é tudo. Dentre os 256 caracteres disponíveis, seu computador possui dois conjuntos de caracteres especiais, utilizados amplamente nos idiomas português e espanhol. E, para maior comodidade do usuário, o TK90X dispõe ainda de uma rotina residente na ROM que possibilita a criação de caracteres inéditos.

A seguir, veremos como ter acesso a todos esses recursos.

18.1. CHR$ NÚMERO

Esta função vem normalmente precedida do comando PRINT e é empregada para se obter o caractere correspondente ao número que lhe serve de argumento. Digite, por exemplo:

```
PRINT CHR$ 38
```

O computador apresentará

```
&
```

Se você verificar a lista de códigos do Apêndice D, verá que o símbolo & corresponde ao número 38.

```
CHR$ 35 = #  CHR$ 36 = $
CHR$ 37 = %  CHR$ 38 = &
CHR$ 39 = '  CHR$ 40 = (
```

O programa seguinte gera uma lista de caracteres (de 32 a 255). Os primeiros (de 0 a 31) são "invisíveis", pois apenas desempenham funções:

```
 5 FOR x=32 TO 255
10 PRINT,,"CHR$";x;" = ";CHR$ x
20 NEXT x
```

18.2. CODE "string"

A função CODE se aplica a um string, retornando o código do primeiro caractere da seqüência alfanumérica.

```
PRINT CODE "&abcdef"
```

Fornece 38 como resultado. Você já viu que 38 é o código ASCII do caractere & , que é o primeiro da seqüência do exemplo.

18.3. Caracteres do Modo Gráfico

O modo gráfico **G** dá acesso aos padrões gráficos inscritos nas teclas 1 a 7. Para obtê-los, basta digitar (em modo **K**):

PRINT "

Mudar para modo gráfico e pressionar a tecla correspondente ao desenho nela inscrito; retornar ao modo **L** e fechar as aspas.

Se for acionado <CAPS SHIFT> ou <SYMBOL SHIFT>, obtém-se o inverso do desenho, somando-se, assim, 16 caracteres gráficos diferentes.

18.4. UDG

A função UDG é o meio pelo qual são acessados os dois conjuntos de caracteres especiais do português e do espanhol.

Antes de mais nada, digite as letras de A a U, em modo gráfico (comece com PRINT ", em modo **K**, e feche as aspas no modo **L**):

PRINT"ABCDEFGHIJKLMNOPQRSTU"

O resultado será a impressão dessas letras em tipo maiúsculo.

Não seria nada interessante ser o conjunto obtido em **G** idêntico ao conseguido por meio de <CAPS SHIFT>. Na verdade estas letras permitem sua substituição por outros símbolos gráficos que você mesmo venha a criar, conforme veremos adiante.

Antes de criar novos caracteres, vamos acessar os caracteres especiais da língua portuguesa. Digite:

UDG Ø <ENTER> (obtém-se UDG com <SYMBOL SHIFT> e <X>)

Agora repita PRINT "; entre em Modo **G** e digite novamente cada letra de A a U (deixe um espaço entre elas para melhorar o visual). Desta vez o resultado muda bastante:

Á É Í Ó Ú á é í ó ú À à Â Ê â ê
ô ã õ Ç ç

Para acessar o conjunto de caracteres do espanhol, digite UDG 1. Repita a operação PRINT e você terá na tela:

Á É Í Ó Ú á é í ó ú Ô ñ Ñ Ã Õ Ü ü

Nota: Os dez primeiros caracteres são comuns aos dois conjuntos acima.

Agora você pode escrever corretamente português ou español. Não se preocupe se ao listar o seu programa, os caracteres saírem trocados; o importante é o resultado.

Entre UDG Ø e programe:

1Ø UDG Ø: PRINT "Português" (para obter ê use p no modo gráfico)

Antes de digitar a próxima linha, entre:

UDG 1

Em seguida:

20 UDG 1: PRINT "Español" (para obter ñ use L no modo gráfico)

No momento em que pressionar <ENTER>, a listagem sairá confusa:

10 UDG 0: PRINT "Portuguüs" (o ê saiu como ü porque mudou para UDG1)

20 UDG 1: PRINT "Español"

Mas, rodando o programa, tudo sairá conforme o planejado:

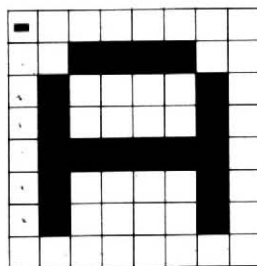
Português
Español

Antes de seguir as instruções do próximo item, desligue e religue seu computador para que ele tire da RAM o código UDG 1 (ou UDG 0). O comando NEW não é suficiente para fazê-lo, pois esta parte da memória está protegida (falaremos sobre a memória do computador nos capítulos finais do manual).

18.5. Definindo Caracteres

Vamos ampliar o manuseio da função UDG.

Digite: <UDG> <2> <ENTER>

Você acaba de habilitar uma função utilitária. Na sua tela você verá uma matriz de 8 por 8 elementos, contendo a letra A, e, abaixo da matriz, três linhas, como na figura a seguir:

ABCDEFGHIJKLMNOPQRSTU
ABCDEFGHIJKLMNOPQRSTU
A=A

A primeira linha representa as letras (A a U) do teclado, disponíveis para o usuário definir que caractere gráfico cada tecla vai receber. A segunda linha representa o conteúdo atual de cada tecla. Este conteúdo pode ser mudado por você, mas sempre que o computador for ligado, o conteúdo padrão (este da figura) aparecerá.

Se você fizer uma retrospectiva, verá que no item UDG obteve o conteúdo padrão das teclas A a U em modo **G**, quando digitou:

<PRINT> <"> **G** <A> <B> <C>... **L** <">

Não confunda com os caracteres que obteve com UDG Ø e UDG 1. Aqueles não devem ser modificados, a menos que seja necessário.

A terceira linha representa o caractere atual, isto é, a tecla que está sendo manipulada: neste caso, a tecla A. Faça um teste: para mudar a tecla atual para U, por exemplo, apenas tecle <U>.

A matriz agora mostra a "estrutura" ampliada 8 vezes do caractere U, e a terceira linha mostra o U como tecla atual. Esta terceira linha mostra o caractere definido no tamanho real.

Note que a matriz tem o fundo na cor branca e os quadrinhos preenchidos em preto. São as cores-padrão para INK e PAPER.

Se você determinar (antes de entrar em UDG 2), por exemplo, INK 2 e PAPER 6, obterá a matriz e o caractere nela contido, na cor (2) vermelho; o fundo da tela em amarelo (6) (cor do PAPER), e as três linhas abaixo da matriz nas cores padrão, ou seja, em preto sobre fundo branco. Estas linhas estarão sempre nestas cores.

Suponha que você queira modificar o conteúdo da tecla O. A primeira coisa a fazer é colocar esta tecla em evidência, teclando <O>. Isto feito, a estrutura de O, aparecerá na matriz, e, na terceira linha, você terá O=O. Mas você não quer O=O. Vamos supor que você queira que na tecla O, em modo **G**, apareça o sinal ≈. Terá que criá-lo, pois ele não existe no computador:

Primeiramente limpe a matriz, para iniciar a construção da estrutura de ≈:

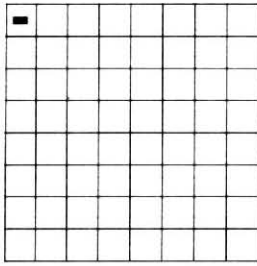
Mantendo pressionada <CAPS SHIFT> ou <SYMBOL SHIFT> digite <1>.

Três coisas acontecem:

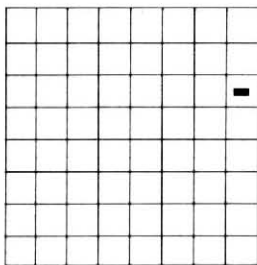
A matriz fica vazia, com exceção de um pequeno ponto.

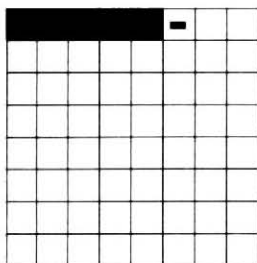
Ao lado de O=, na terceira linha, não há mais nada.

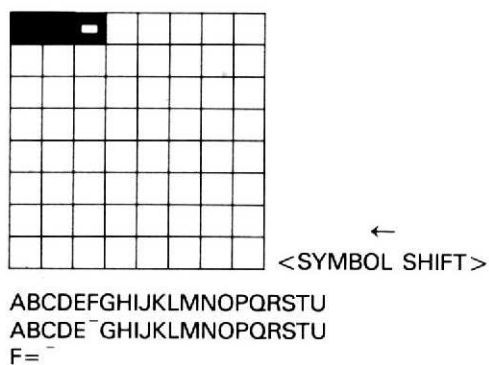
Há um espaço em branco na segunda linha, informando que a tecla O não tem representação gráfica neste momento.

Dissemos que na matriz permaneceu um ponto após seu esvaziamento. Este ponto é o cursor de gráfico: um indicador da posição em que a matriz será preenchida ou apagada.

ABCDEFGHIJKLMNOPQRSTU
ABCDE GHIJKLMNOPQRSTU
F=

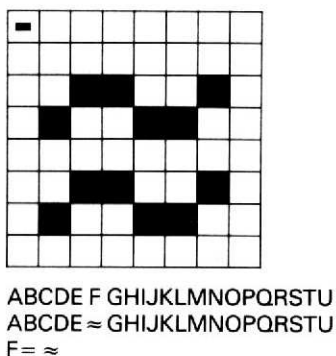
Para mover o cursor dentro dos limites da matriz (ele jamais ultrapassará estes limites), use as setas de direção.

Pressione apenas as teclas 5 a 8, sem <CAPS SHIFT>.

ABCDEFGHIJKLMNOPQRSTU
ABCDE GHIJKLMNOPQRSTU
F=

Se, no entanto, você utilizar as setas, mantendo <CAPS SHIFT> pressionada, moverá o cursor, preenchendo o quadrado onde ele se encontra, com a cor determinada anteriormente por INK. Caso não tenha sido determinada, como foi dito, a cor será o preto (Ø).

ABCDEFGHIJKLMNOPQRSTU
ABCDE¯GHIJKLMNOPQRSTU
F=¯

Para apagar os quadrinhos já preenchidos, use o mesmo sistema de setas, mantendo, porém, a tecla <SYMBOL SHIFT> pressionada.

Note que, ao mesmo tempo em que ocorrem mudanças na matriz, elas vão aparecendo ao lado da letra O, na terceira linha, e na posição desta letra na segunda linha. Você pode assim ter uma noção de proporção em relação aos outros caracteres.

Agora que já conhece os princípios de criação, monte na matriz a estrutura para ≈.
Siga o modelo da figura a seguir, ou melhore-o se quiser:

ABCDE F GHIJKLMNOPQRSTU
ABCDE ≈ GHIJKLMNOPQRSTU
F= ≈

Para sair da rotina UDG 2, pressione <CAPS SHIFT> e <Ø>.

De agora em diante você obterá sempre o caractere (ou caracteres) que criou em UDG 2. Para isto, basta apertar a tecla correspondente em Modo **G**.

Use o símbolo ≈, que acabamos de construir, aproveitando-o no programa:

```
<CLS>
1Ø FOR r=Ø TO 233
2Ø PRINT " ≈ ";:REM usando a letra "F" em modo gráfico
3Ø NEXT r
<RUN>
```

O resultado deverá ser uma tela preenchida com os novos símbolos

Notas: Utilize a rotina UDG 2 logo após o computador ter sido ligado, ou antes de ter utilizado os comandos UDG Ø e/ou UDG 1. Após a leitura de um destes códigos pela RAM, você não terá mais acesso ao código livre para definições. A rotina UDG 2 também funciona neste caso, mas traz para a RAM os caracteres do último código acessado.

Você verá no Capítulo 23 como guardar os símbolos gráficos que eventualmente tenha criado, para poder usá-los quando for necessário.

18.6 SCREEN$

O formato da função SCREEN$ é o seguinte:

SCREEN$ (l, c)

O primeiro argumento (l) determina a linha e o segundo (c) a coluna nas quais atuará a função.

SCREEN$ é utilizada para a identificação do caractere que ocupa determinada posição da tela, ou se o caractere presente for do tipo determinado pelo usuário, a função retornará um string vazio.

MANIPULANDO A MEMÓRIA

19. MANIPULANDO A MEMÓRIA

19.1. POKE endereço dado

A instrução POKE serve para introduzir dados diretamente na memória do computador. Lembre-se de que os dados somente poderão ser introduzidos na memória RAM, uma vez que o conteúdo da ROM é inalterável. Para conseguir tal entrada você deverá indicar:

1) O endereço da memória RAM ao qual deseja enviar o dado.
2) O dado a ser introduzido

Exemplo:

```
POKE 23609,20
     decimal  decimal
```

POKE 23609,20 indica a entrada do valor 20 no endereço 23609. Estes números são decimais, porém, serão armazenados pelo computador no formato binário.

Os dados são armazenados em bytes (8 bits). Assim sendo, o dado a ser introduzido diretamente na memória deve estar compreendido entre 0 e 255. Se for tentada a introdução de um número fora destes limites, o computador apresentará a mensagem:

B Inteiro excede limite

A mesma mensagem será apresentada se o endereço for maior que 65535, dado que este é o endereço limite da memória considerada pela CPU do TK90X.

Caso seu computador tenha 16k de RAM, POKE não atuará em endereços acima de 32767.

19.2. PEEK endereço

PEEK efetua a leitura de um determinado endereço da memória retornando o dado que ali se encontra:

Exemplo:

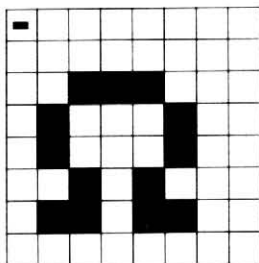
```
10 LET K=PEEK 23609
20 PRINT K
```

Executado o programa, o conteúdo do endereço 23609 é apresentado na tela (no caso de você haver introduzido 20, este é o resultado do PEEK).

19.3. BIN

Dissemos que cada dado é composto por 8 bits, ou seja, num byte. Para representar, por exemplo, um caractere, teríamos que "traduzir" sua forma em bits. O computador usa a linguagem binária e só entende os números 0 e 1. Se o bit está com o valor 0, está desligado; com o valor 1, ligado.

Para entender melhor o que foi explicado, verifique a composição dos caracteres no capítulo anterior. A matriz de 8 por 8, que você conheceu com a função UDG2, serve perfeitamente para ilustrar um byte e seu conteúdo.

A figura adiante completará a explicação:

Associa-se a cada quadrinho preenchido o valor 1 e, a cada vazio, o valor zero.

Para representar a seqüência toda, teríamos, portanto:

```
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 1 1 1 0 0 0
0 1 0 0 0 1 0 0
0 1 0 0 0 1 0 0
0 0 1 0 1 0 0 0
0 1 1 0 1 1 0 0
0 0 0 0 0 0 0 0
```

Você não pode, todavia, introduzir esta seqüência na forma em que ela se apresenta, pois está trabalhando em BASIC, e no sistema decimal. Para utilizar o POKE, teríamos que obter tais números em decimal, conforme foi dito no início do capítulo. Para tanto, utiliza-se a função BIN, que tem o seguinte formato:

BIN seqüência binária

Exemplo:

```
PRINT BIN 00011100
```

O computador retornará:

```
28
```

ou

```
POKE 23609, BIN 00011100
```

O computador introduzirá 28 no endereço especificado (23609).

Perceba que este POKE faz com que o computador emita som ao pressionar-se qualquer tecla. Se você incrementar o valor, aumentará a duração do som.

Para introduzir o byte representado pela matriz da figura anterior, teríamos que transformar em decimal toda a seqüência binária, colocando a função BIN antes de cada conjunto de oito bits. Mas, que endereço usaríamos para o POKE?

Vamos associar o "caractere" que está na matriz a uma letra do teclado: A, por exemplo.

Em seguida utilizaremos a função USR, tendo A como argumento. Significa que o "caractere" que criarmos será armazenado a partir do endereço A (USR "A", USR "A"+1, USR "A"+2 ... USR "A"+7).

Nota: A função USR é usada aqui para converter no correspondente caractere gráfico a ser alterado um argumento tipo string, alocado no endereço do primeiro byte na memória.

Execute o programa:

```
<NEW>
10 FOR x=0 TO 7
15 READ s
20 POKE USR "a"+x,s
25 NEXT x
30 DATA 0,0
35 DATA BIN 00111000,BIN 01000100
40 DATA BIN 01000100,BIN 00101000
45 DATA BIN 01101100,0
```

Nota: Uma seqüência de zeros em binário pode ser representada diretamente por um zero, em decimal, sem ser precedido por BIN (linhas 30 e 45).

Para confirmar que o novo "caractere" foi introduzido na RAM, entre em modo gráfico e use A . Se quiser ver melhor ainda, entre em UDG2 e coloque "A" em evidência.

A função USR pode vir seguida de um número como argumento. O uso deste formato será visto posteriormente, no capítulo 25.

GRÁFICOS

20. GRÁFICOS

20.1. PLOT x,y

Esta instrução tem como objetivo plotar (colocar um ponto) em qualquer local especificado da tela.

PLOT usa coordenadas de (0,0) a (255,175) oferecendo, portanto, 45.056 pontos na tela, o que proporciona uma melhor definição de gráficos e de desenhos.

Caso os limites admissíveis sejam ultrapassados, haverá uma mensagem de erro:

VÁLIDO	INVÁLIDO
PLOT 255,175	PLOT 255,255

Entre a seguinte instrução:

```
PLOT INK 1; 127,87
```

Observa-se um ponto azul no centro da tela.

INK 1 definiu a cor do ponto
(127,87) definição da coluna, linha

As coordenadas (0,0) localizam-se no canto inferior esquerdo da tela, não sendo, portanto, coincidentes com as coordenadas (0,0) do PRINT. Introduza e execute o programa.

```
<NEW>
10 FOR x=0 TO 255
20 PLOT INK 2; x,0
30 NEXT x
```

Note que a reta tem seu início no ponto inferior esquerdo e estende-se até a coluna 255 (valor máximo de x).

20.2. DRAW x,y

Desenha um ponto ou uma seqüência de pontos a partir da coordenada definida pelo último PLOT (ou DRAW). Caso PLOT (ou DRAW) não tenham sido utilizados anteriormente, o ponto inicial de DRAW será (0,0).

Exemplo:

```
<NEW>
10 DRAW 80,80
<RUN>
```

O ponto inicial antes de se usar DRAW era (0,0).

DRAW incrementou 80 pontos, para o primeiro parâmetro, e 80, para o segundo, formando, assim, uma transversal.

Acrescente esta linha e rode o programa:

```
20 DRAW 80,0
```

DRAW agora incrementou 80 pontos para o primeiro parâmetro (coluna) e manteve o segundo parâmetro (linha) do DRAW anterior.

DRAW também restringe ambos os parâmetros, desde que os valores atribuídos a eles sejam negativos.

Acrescente esta linha:

```
30 DRAW - 80,-80
40 DRAW - 80,0
<RUN>
```

As coordenadas especificadas não devem ultrapassar os limites (0,0) a (255,175), caso contrário, ocorrerá uma mensagem de erro.

20.2.1. DRAW x,y,z

Pode-se usar um terceiro parâmetro em DRAW, com a finalidade de traçar um arco com medidas em radianos. Se o valor deste parâmetro for positivo, o arco será descrito no sentido anti-horário, se for negativo, no sentido horário.

Exemplo:

```
<NEW>
10 PLOT 127,87
20 DRAW 0,20,5
30 DRAW 0,-20,-4
```

A linha 10 plota um ponto no centro da tela, portanto, o ponto inicial do primeiro DRAW será neste mesmo local (127,87).

A linha 20 traça um arco a partir da última coordenada, com o terceiro parâmetro definindo o grau de abertura do arco em radianos.

A linha 30 também traça um arco a partir da última coordenada, porém com o terceiro parâmetro negativo, fazendo o arco no sentido horário.

Obs.: O terceiro parâmetro mostra a fração de circunferência que deve ser traçada. Uma circunferência completa mede 2 radianos. Se este parâmetro for igual a π, será desenhada a metade de uma circunferência; 0.5*PI corresponderá a 1/4 de circunferência, e assim, sucessivamente.

20.3. POINT x,y

Indica a situação de um ponto especificado pelas coordenadas x,y.

Se o resultado for **0**, então o ponto na coordenada definida tem a mesma cor do fundo (tela). Se o resultado for **1**, o ponto tem a cor da tinta (caractere com INK definido).

Exemplo:

```
<NEW>
10 PRINT POINT (0,0)
20 PLOT 127,87
30 PRINT POINT (127,87)
```

20.4. CIRCLE x,y,r

Desenha uma circunferência inteira, desde que sejam atribuídas as coordenadas específicas do centro e o raio.

Exemplo:

```
CIRCLE 127,87,20
```

As instruções PLOT, DRAW e CIRCLE podem ser usadas em conjunto com as instruções INK e PAPER, para a definição de cor. Exemplo:

```
<NEW>
10 PLOT INK 1; 127,87
20 CIRCLE INK 2; 127,87,40
30 DRAW INK 3; 50,-20
```

Experimente usar PAPER em lugar de INK.

Caso você queira aplicar INVERSE ou FLASH nestas instruções, use-as do mesmo modo que INK e PAPER, porém, seguindo suas regras:

```
INVERSE 0 ou INVERSE 1

  FLASH 0 ou FLASH 1
```

TEMPO E MOVIMENTO

21. TEMPO E MOVIMENTO

21.1. PAUSE n

Freqüentemente é preciso interromper temporariamente um programa ou uma rotina de programa. Para este fim usa-se o comando PAUSE.

Exemplo:

```
30 PAUSE 400
```

n pode ser um número inteiro até 65535, o que ocasiona cerca de 18 minutos de paralisação. Se n = 0, a pausa é, teoricamente, infinita.

O tempo de duração deve ser obtido da seguinte forma:

$$t = \frac{n}{q}$$

- t = tempo em segundos
- n = parâmetro definido para PAUSE
- q = é igual a 60 em países onde a freqüência da rede é de 60 Hz, e é igual a 50 onde a freqüência é de 50 Hz

Pode-se interromper a execução do comando mesmo antes do tempo especificado. Basta que se pressione qualquer tecla, no decorrer da paralisação (exceto as teclas <SHIFT>).

Exemplo:

```
<PAUSE> 1000 <ENTER>
```

Pressione qualquer tecla para interromper.

Agora rode este programa:

```
<NEW>
10 FOR x=1 TO 10
20 PRINT AT 10,14;"((";x;"))"
40 PRINT PAPER 4;AT 2,12;"p a u s e"
45 PAUSE 80
50 CLS
60 NEXT x
```

Note como o tempo é controlado pela linha 45. Pause n equivale a um tempo como o especificado acima, porém, o processamento também toma tempo. No caso da linha 45, há uma paralisação de um pouco mais de um segundo (PAUSE 80).

21.2. INKEY$

A função INKEY$ lê o teclado e verifica qual tecla foi acionada. Caso o teclado não tenha sido acionado, INKEY$ é igual a um string vazio ("").

A função INKEY$ é similar ao comando INPUT, porém com duas diferenças marcantes:

A função INPUT permite a entrada de vários caracteres, que são considerados, em conjunto, como o dado requisitado. A função INKEY$ permite a entrada de apenas 1 caractere, e somente este caractere representa o dado necessário a ser inserido.

Na função INKEY$, ao contrário de INPUT, não se digita <RETURN> na introdução do dado, pois esta é feita automaticamente.

Exemplo:

```
<NEW>
  10 PRINT INK 2,AT 2,2;"quer rodar o programa ? (s/n)"
  20 IF INKEY$=""THEN GOTO 20
  30 IF INKEY$="n"THEN GOTO 200
  35 IF INKEY$< >"s" AND INKEY$< >"n"THEN GOTO 20
  40 CLS
  45 PRINT PAPER 5;AT 2,0;"INKEY$";"recebe apenas um caractere"
  50 PRINT PAPER 4;AT 5,0;"numerico ou alfanumerico"
  60 STOP
 200 CLS
 210 PRINT INK 3;AT 10,10;" entao e o fim"
```

A linha 20 aguarda que o teclado seja acionado. Caso isto não ocorra, haverá um desvio para a mesma linha.

As linhas 30 e 35 impõem uma condição: a tecla a ser pressionada deve ser <s> ou <n>. De outra forma, haverá novamente o desvio para linha 20.

Os desvios serão feitos de acordo com a condição do INKEY$. Se <n>, então desvio para linha 200; se <s>, o programa não será desviado, executando-se assim as linhas subseqüentes.

EFEITOS SONOROS

23. LEITURA E GRAVAÇÃO DE FITAS (Sistema Easy-load)

23.1. EASY-LOAD

O TK90X é dotado do sistema "easy-load" para armazenagem de informações em fita magnética. Este sistema é sensivelmente superior a tudo o que existe em matéria de uso de fitas cassete com computadores. Além de um "feedback" (retorno) visual do andamento da operação de carga, ele permite a indexação de seus arquivos, com pesquisa automática pelo nome, e ainda é quase independente do ajuste de volume do gravador, desde que se usem equipamentos e fitas de boa qualidade.

23.2. SAVE "programa"

Quando o TK90X é desligado, perde-se todo o programa e variáveis armazenados na memória. Uma das maneiras de se preservar programas é instruir o computador para gravá-los em fita cassete, de modo que possam posteriormente ser recuperados.

A instrução SAVE tem a finalidade de gravar programas e variáveis definidas.

Esta instrução é seguida por um string que é fornecido pelo usuário, ou seja, o nome do programa.

Procedimento para gravação:

Digite este programa:

```
<NEW>
10 FOR x=0 TO 154 STEP 4
20 FOR z=50 TO 100
25 PLOT INK 4;x,z
30 NEXT z
35 DRAW z,5,3
36 DRAW INK 2;-z,5,3
50 NEXT x
```

Conecte o cabo de leitura/gravação nas tomadas MIC do gravador e MIC do micro.

Rebobine a fita até passar a parte não magnetizada.

Digite:

<SAVE>"programa 1" (o nome deve ter no mínimo 1 caractere)

Tecle: <ENTER>

Surgirá na tela a mensagem:

LIGUE O GRAVADOR E TECLE <ENTER>

Lembre-se de que o gravador deve estar ligado no modo de gravação (PLAY/REC).

Após ser teclado <ENTER>, surgirão nas bordas da tela faixas horizontais e serão emitidos ruídos agudos. Ao primeiro sinal, ocorre a gravação do nome do programa; ao segundo, quando as faixas são mais estreitas e nas cores azul e amarelo, o próprio programa está sendo gravado.

Espere até que apareça no vídeo a mensagem:

```
Ø Executado Ø:1
```

Pare o gravador: o programa está gravado.

23.3. LOAD "programa"

Esta instrução recupera os programas gravados em fita pela instrução SAVE. O procedimento da instrução LOAD é o seguinte:

É feita uma pesquisa na fita. Caso existam outros programas gravados antes do que deve ser carregado, surgirão na tela seus respectivos nomes. Quando o programa desejado é encontrado, começa a ser transferido para a memória do computador.

O nome que estiver entre aspas deverá ser idêntico ao do programa desejado; caso contrário, o sistema não encontrará o programa.

Pode-se também dispensar o nome entre as aspas:

```
LOAD" "
```

Neste caso o primeiro programa encontrado será transferido para a memória.

Ao executar-se a instrução LOAD, observam-se, nas bordas da tela de TV, finas faixas horizontais. Acompanha-as um sinal sonoro agudo e constante.

Os efeitos visuais e sonoros da leitura da fita são semelhantes aos da gravação.

Para que se possa fazer uma transferência do programa da fita para o computador deve-se:

Conectar o cabo de gravação/leitura nas tomadas EAR do gravador e EAR do micro.

Ajustar o volume de som do gravador para o máximo agudo.

Acionar o comando LOAD" " ou LOAD "nome".

Teclar <ENTER>.

Acionar PLAY no gravador.

Aguardar a mensagem: Ø Executado Ø:1.

23.4. SAVE ... LINE

Estas duas instruções conjugadas têm a mesma finalidade da instrução SAVE, porém, quando termina a leitura do programa pela instrução LOAD, este é imediatamente executado a partir do número da linha especificado após LINE.

Procedimentos para a gravação no modo SAVE/LINE:

Define-se:

```
SAVE "nome do programa" LINE número da linha
```

Seguem-se as instruções do modo SAVE vistas anteriormente.

23.5. SAVE ... DATA

Pode-se também guardar dados de uma matriz em fita cassete. Isto é feito através das duas instruções conjugadas: SAVE e DATA.

Exemplo:

```
SAVE"nome do programa" DATA nome da matriz ()
```

O nome da matriz indica qual matriz se quer gravar. Poderá ser de apenas uma letra ou de uma letra seguida por $ (no caso de matriz alfanumérica), sendo obrigatório o uso dos parênteses no final; caso contrário, ocorrerá erro de sintaxe.

Segue abaixo um exemplo onde SAVE/DATA está em linha de programa (nada impede que sejam usadas no modo imediato).

Prepare o gravador para a gravação e execute:

```
<NEW>
10 DIM a$(5,10)
20 FOR x=1 TO 5
30 INPUT "ENTRE NOME !";a$(x)
35 NEXT x
40 SAVE "teste" DATA a$()
```

A linha 10 do programa estabelece que poderão ser inscritos na matriz cinco strings, com dez caracteres cada um.

Para facilitar a entrada de dados, foi utilizado um "loop" FOR...NEXT, de cinco ciclos, que correspondem à quantidade de dados que serão introduzidos em a$.

A linha 40 grava todos os dados de a$(1) até a$(5). Porém, não grava o programa, somente os nomes introduzidos pelo INPUT da linha 30. Caso se queira gravar o programa e o conteúdo da matriz, deve-se usar SAVE"nome" no formato normal).

23.6. VERIFY ... DATA

Após ter-se gravado os dados de uma matriz usando as instruções SAVE DATA, pode-se comparar se o que foi gravado coincide com o que ainda permanece na memória. VERIFY DATA tem essa finalidade.

Exemplo:

Volte a fita ao ponto onde começou a gravação dos dados da matriz.

Conecte o cabo de leitura/gravação: nas tomadas EAR (micro) e EAR(gravador).

Ajuste o volume.

Entre:

```
VERIFY "teste" DATA a$()
```

Serão apresentados na tela o nome atribuído à matriz e as faixas horizontais características. Um sinal sonoro agudo constante será emitido.

Aguarde a mensagem:

```
Ø Executado Ø:1 ou

R Erro de leitura (se a gravação não foi satisfatória, devendo, ser repetida)
```

A instrução VERIFY foi usada para verificar gravações de dados de matrizes, mas pode também ser usada com outros tipos de SAVE. Verifique na tabela de instruções.

23.7. LOAD ... DATA

Tendo sido gravados os dados de uma matriz, pode-se recuperá-los usando as instruções conjugadas LOAD/DATA. O procedimento de leitura é idêntico ao do LOAD normal, visto anteriormente.

O formato da instrução é:

```
LOAD"teste" DATA a$( )
```

Limpe a memória:

```
<NEW>
```

Digite o programa a seguir:

```
1Ø REM este programa recupera as matrizes
2Ø PRINT "prepare o gravador para leitura"
25 PRINT AT 4,2: "quando OK tecle algo"
3Ø IF INKEY$ = "" THEN GOTO 3Ø
4Ø LOAD "teste" DATA a$()
45 CLS
5Ø FOR c=1 TO 5
6Ø PRINT a$(c)
7Ø NEXT c
```

Rode o programa.

LOAD ... DATA pode ser usado em modo imediato (sem número de linha).

Nota: Lembre-se de que quando se carrega uma matriz de dados alfanuméricos, apaga-se qualquer matriz ou qualquer cadeia simples com o mesmo nome, caso haja esta coincidência.

23.8. SAVE ... CODE

SAVE/CODE possibilita o armazenamento de bytes em fita. Os bytes gravados podem representar uma forma gráfica gerada por um programa, instruções dadas em modo imediato ou qualquer caractere que esteja na tela no momento do comando.

O formato da instrução SAVE CODE é:

SAVE"nome do programa"CODE endereço, n.º de bytes"

O programa a seguir define formas em alta-resolução, utilizando todas as cores de que dispõe o TK90X. No final da execução do programa grava somente a tela, usando SAVE/CODE.

```
<NEW>
 5 LET a=40: LET f=0: LET k=0
 6 LET t=50
10 FOR x=k TO 10+k
20 FOR z=t TO a
25 PLOT INK f;x,z
30 NEXT z
35 LET a=a+8
40 NEXT x
41 LET f=f+1: IF f=7 THEN LET f=0
45 LET a=60: LET k=k+15
50 LET t=t-5: IF t=5 THEN LET t=50
55 IF x>=240 THEN GOTO 70
60 GOTO 10
70 PRINT AT 2,2;"tecle algo para gravar"
80 IF INKEY$="" THEN GOTO 80
85 PRINT AT 2,2;"  observe o desenho   "
90 SAVE "tela"CODE 16384,6912
```

Observe que, após o término do desenho, uma ordem de gravação é executada assim que seja autorizada (linha 70), sendo tudo o que estiver na tela (somente na tela) gravado. Para tanto, foi preciso definir o endereço (16384) e a quantidade de bytes que se quer gravar. No caso, 16384 corresponde ao endereço do primeiro byte do arquivo do display (imagem da TV), e 6912 é o número de bytes existentes no arquivo.

Os endereços e quantidades de bytes são definidos conforme a necessidade do usuário, respeitados os limites admissíveis.

Para armazenar, por exemplo, a área de caracteres gráficos definíveis pelo usuário, você pode usar:

SAVE "graficos" CODE USR "a", 21*8

23.9. LOAD ... CODE

Esta instrução recupera as informações gravadas por SAVE/CODE: imagens, desenhos, gráficos e programas em linguagem de máquina.

A instrução tem os seguintes formatos:

```
LOAD"tela"CODE 16384
     ou
LOAD"tela"CODE
```

Quando o endereço em que se quer carregar os dados da fita é diferente do original, usa-se o primeiro formato. Caso o endereço e tamanho sejam mantidos, a posição e tamanho originais serão assumidos.

Exemplo:

```
LOAD "tela" CODE 16384, 5000
```

23.10. LOAD ... SCREEN$

Para facilitar o processo de leitura a ser operado por LOAD/CODE, usa-se a instrução LOAD SCREEN$. Assim, em vez de se definirem os valores do primeiro byte e a quantidade de bytes, emprega-se a instrução SCREEN$, equivalente aos códigos a serem definidos no SAVE.

Formato:

```
LOAD"tela"SCREEN$
```

A gravação (SAVE/CODE) também pode ser alterada para SAVE"tela"SCREEN$, sem necessidade da definição dos valores de endereço e quantidade de bytes.

Formato:

```
SAVE"tela"SCREEN$
```

23.11. MERGE

Possibilita a carga de um programa armazenado em fita, sobrepondo-o ao que já se encontra na memória. Caso exista a coincidência de linhas dos programas, prevalecerá a linha recém-carregada, devendo-se, portanto, tomar muito cuidado para que não ocorra a perda de uma linha importante do programa.

A mensagem 4 Memória lotada ocorrerá caso se tente carregar mais programas do que comporta a memória do computador. Se isso ocorrer, grave o programa que se encontra na memória, mesmo que incompleto, e limpe-a.

Procure reduzir todo o conjunto de instruções para poder obter uma fusão definitiva e completa das linhas de programação.

A instrução MERGE é utilíssima para recuperarem-se programas longos que foram gravados por partes. MERGE reúne todas as partes, formando um só programa.

Exemplo: Digite este programa

```
<NEW>
  1 LET a=0: LET k=0: LET l=0
  4 CLS
  7 FOR c=0 TO 120 STEP 20
 10 FOR x=0 TO 10
 20 PLOT c+x,140
 30 DRAW INK a;0,10
 35 SOUND .05,x
 40 NEXT x
 45 LET a=a+1
 50 NEXT c
 53 IF k=1 THEN GOTO 77
 55 PRINT "prepare o gravador": PRINT "para gravar este programa"
 60 PRINT "tecle algo se tudo ok"
 65 IF INKEY$="" THEN GOTO 65
 75 SAVE "prog1" LINE 5
 77 IF l=1 THEN GOTO 83
<RUN>
```

Após ter seguido as instruções do programa, e tê-lo gravado em fita, limpe-o da memória:

```
<NEW>
```

Agora digite este outro programa:

```
 80 LET k=0
 83 LET a=0
 85 FOR r=0 TO 120 STEP 20
 90 FOR x=0 TO 20
 95 PLOT x,120-r
100 DRAW INK a;x,10
105 SOUND .05,x+10
110 NEXT x
115 LET a=a+1
120 NEXT r
125 INK 4
128 IF k=1 THEN STOP
130 PRINT "prepare o gravador"
135 PRINT "para ler o programa anterior"
140 PRINT "e tecle algo quando ok"
145 IF INKEY$="" THEN GOTO 145
190 MERGE "prog1"
195 LET a=0: LET l=1: LET k=1: GOTO 4
```

Antes de rodar este programa, volte a fita ao início do programa gravado anteriormente (1.º programa). Logo em seguida entre:

```
<RUN>
```

23.11. Tabela de Instruções

Segue abaixo a tabela de instruções vistas neste capítulo:

INSTRUÇÕES	FORMATO	FINALIDADE
SAVE	SAVE "nome"	Armazena em fita magnética um programa que se encontra na RAM.
SAVE LINE	SAVE "nome" LINE n	O mesmo que SAVE, porém determina a execução automática do programa a partir da linha n, logo após sua leitura.
SAVE DATA	SAVE "nome" DATAx()	Armazena em fita magnética os dados de uma matriz numérica X ou alfanumérica X$.
SAVE CODE	1-SAVE "nome" CODEn1,n2 2-SAVE "tela" CODE 16384,	1-Salva n2 bytes começando no endereço n1. 2-Salva imagem do dispay com endereço do primeiro byte do display (16384) e quantidade de bytes que irá gravar (6912).
SAVE SCREEN$	SAVE "nome" SCREEN$	O mesmo que SAVE CODE, apenas tendo SCREEN$ representando o endereço e tamanho padrões (16384,6912).
LOAD	LOAD "nome"	Carrega um programa gravado em fita magnética para a memória RAM do computador.
LOAD DATA	LOAD "nome" DATA...()	Carrega de uma fita para a RAM dados de uma matriz numérica x ou alfanumérica x$.
LOAD CODE	LOAD "nome" CODE n1,n2 ou LOAD "nome" CODE	Carrega um programa gravado por SAVE/CODE num endereço especificado ou não. Não é necessário que se defina a quantidade de bytes.
LOAD SCREEN$	LOAD "nome" SCREEN$	O mesmo que LOAD/CODE, apenas utilizando o endereço e tamanho da área de video em forma implícita.
MERGE	MERGE "nome"	Executa a sobreposição de um programa gravado em fita num outro que já se encontra na memória.
VERIFY	VERIFY "nome"	Compara a gravação de um programa com o programa original que ainda se encontra na memória RAM.
VERIFY DATA	VERIFY "nome" DATA	O mesmo que VERIFY, porém comparando dados de uma matriz gravados em fita, com dados da matriz que ainda está na memória.
VERIFY CODE	VERIFY "nome" CODE nl,N2	Compara os blocos de dados armazenado na fita com o correspondente na memória entre os endereços n1 e n1+n2.

MEMÓRIA

24. MEMÓRIA

A unidade central de processamento (UCP ou CPU) de seu computador é o microprocessador Z80.

O Z80 é encarregado de processar todos os dados e informações, utilizando um local intermediário de armazenamento chamado de memória.

A armazenagem de informações na memória é feita sob a forma de bytes, e o lugar que cada byte ocupa, diz-se "endereço".

O Z80 é capaz de manipular até 2 (65536) bytes.

24.1. Tipos de Memória

Quando você liga o seu computador ele começa imediatamente a funcionar, apresentando a tela inicial com o padrão de cores e o nome "TK90X". Isto ocorre porque existe dentro dele um programa "pré-gravado" chamado monitor, encarregado de acatar e executar os seus comandos e programas. Este programa é gravado numa memória permanente, chamado ROM (Read-Only-Memory), ocupando os endereços de 0 a 16.383, e não se desgravando nunca, mesmo quando você desliga o seu computador.

Nos endereços de memória acima de 16.383 existe um outro tipo de memória, a memória RAM (Random-Acess-Memory), que pela sua caracteristica de fabricação é volátil, ou seja, pode ser lida e gravada quantas vezes se quiser, apagando-se quando se desliga o computador.

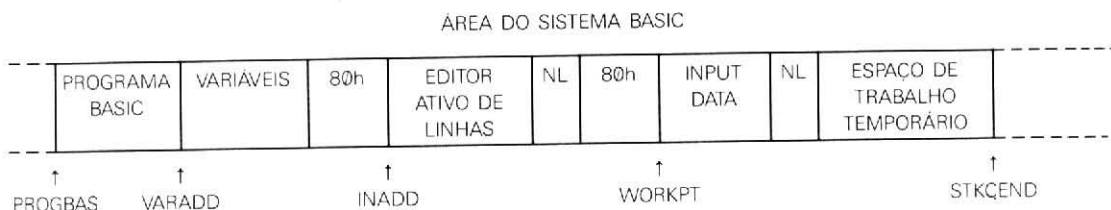
24.2. MAPEAMENTO DA RAM

O Sistema Operacional do TK90X atribui a áreas da RAM diferentes funções, de acordo com o tipo da informação a ser armazenada em cada uma delas.

Abaixo, um diagrama do mapeamento da RAM do TK90X:

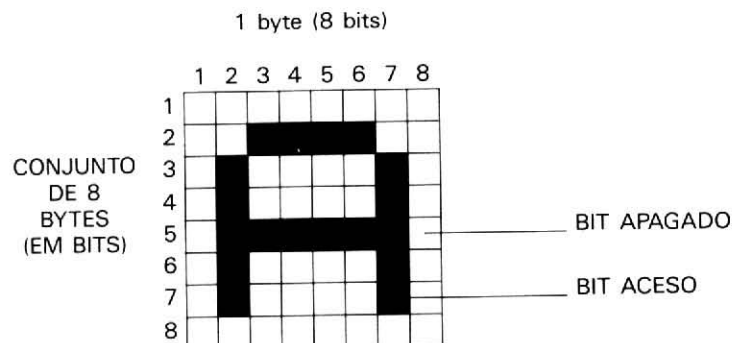
ÁREA DO SISTEMA BASIC

PROGRAMA BASIC	VARIÁVEIS	80h	EDITOR ATIVO DE LINHAS	NL	80h	INPUT DATA	NL	ESPAÇO DE TRABALHO TEMPORÁRIO
↑ PROGBAS	↑ VARADD		↑ INADD			↑ WORKPT		↑ STKÇEND

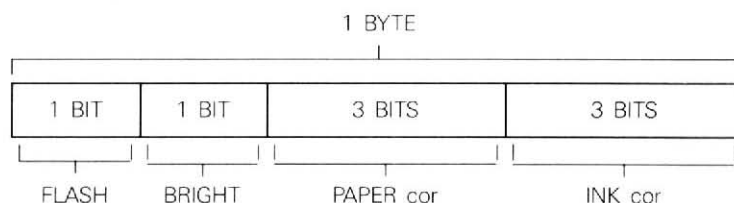
PILHA DE CÁLCULO	ESPAÇO LIVRE	PILHA DE MÁQUINA	PILHA DE GOSUB	?	3EH	DEFINIÇÃO DE GRÁFICOS
↑ STKCEND	↑ ADSPFREE	↑ sp		↑ RAM TOP	↑ UDG 0/1/2	↑ MEMAVLB

24.2.1. Arquivo de Imagem

Existe uma área da RAM, denominada Arquivo de Imagem, que tem como objetivo guardar a imagem presente na te

Cada uma das 768 posições da tela é formada por um quadrado de 8 por 8 pontos, e cada ponto (bit), pode s 1 ou 0 (aceso ou apagado).

Observe a figura a seguir:

O Arquivo de Imagem contém exatamente 6.144 bytes (ou 768 * 8 bits). Como a resolução do vídeo é de 256 X 192, pode-se concluir, então, que cada ponto da tela corresponde a um bit na memória. O bit pode ser ligado ou desligado para compor a imagem desejada.

24.2.2. Atributos

Nesta área de 768 bytes são armazenadas, dentro de cada byte, informações de INK, PAPER, BRIGHT e FLASH de cada uma das 768 posições da tela.

24.2.3. Buffer da Impressora

Área de armazenamento intermediário de todos os caracteres destinados à impressora.

24.2.4. Variáveis do Sistema

Os bytes de memória de 23.552 a 23.733 são reservados para uso exclusivo do sistema. Ao se fazer uma leitura destes endereços, através de PEEK, é possível obterem-se várias informações sobre o sistema.

As variáveis do sistema são representadas por mnemônicos que tornam mais fácil sua identificação. Elas não devem ser confundidas com as variáveis criadas por um programa BASIC. Estão descritas com maiores detalhes no Apêndice C.

24.2.5. Mapa do Speed-drive

O mapa do speed-drive somente é usado com o periférico speed-drive. Normalmente não há nada nele.

24.2.6. Informação do Canal

Esta área contém informação sobre as unidades de entrada e saída de informações a partir do Sistema Operacional.

24.2.7. Área do Sistema BASIC

Área disponível para a entrada de dados e de programas. É flexível à medida em que novos dados sejam inseridos, ou que dados existentes sejam eliminados. Assim:

Havendo na memória um programa no qual seja inserida uma nova linha, é criado espaço para esta pelo deslocamento de tudo o que esteja acima dela, restringindo-se, pois, o espaço livre.

Sendo uma linha eliminada, a área de programação BASIC retrai-se, expandindo-se o espaço livre.

24.2.8. Pilha de Cálculo

A maioria dos números em operação pelo calculador são tratados nesta área.

24.2.9. Pilha de Máquina

É a pilha usada pelo processador Z80 para guardar endereços de retorno e outras informações. Seu ponteiro é o próprio "sp" do microprocessador.

24.2.10. Pilha de GOSUB

Endereço do primeiro item na Pilha de Máquina para retorno após GOSUB.

24.2.11. Definição de Gráficos

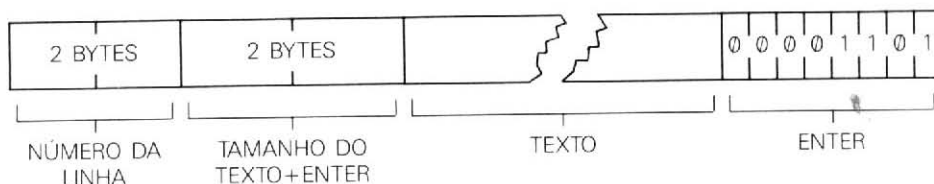
Área reservada para os caracteres acentuados dos ediomas português e castelhano (UDG 0/1) e gráficos definidos pelo usuário (UDG 2).

Esta área poderá ser invadida pelo BASIC em caso de programas muito extensos. Para protegê-la, deve-se utilizar CLEAR endereço, definindo-se assim o maior endereço que o BASIC poderá atingir.

24.3. Linha de Programa em Memória

As linhas de programa seguem um padrão de armazenamento em memória. Por exemplo:

10 PRINT 9 <ENTER>

É retratada da seguinte forma em memória:

Ao contrário de todos os outros casos de números de 2 bytes no Z80, o número da linha é guardado com o seu byte mais significativo primeiro, na ordem pela qual se escreve.

24.4. RAMTOP

Topo da memória RAM.

Quando o TK90X é ligado, faz um teste para verificar a quantidade de memória, colocando a Pilha de Máquina exatamente no topo. É guardado então o endereço do primeiro byte não existente. O comando NEW não atinge as informações que estão acima da RAMTOP, como, por exemplo, os caracteres gráficos definidos pelo usuário, que se encontram nesta posição de memória.

24.5. CLEAR

Pode-se alterar o endereço da RAMTOP utilizando a instrução CLEAR em conjunto com o novo endereço da RAMTOP. Exemplo:

CLEAR 32299 (para versão de 16 K)
ou
CLEAR 65535 (para versão de 48 K)

Resultado:

CLEAR zera todas as variáveis do BASIC e o Arquivo de Imagem.

A posição do PLOT passa a ser 0,0.

Executo um RESTORE, limpa a pilha do GOSUB e, naturalmente, coloca o novo valor de RAMTOP, ficando entre a pilha do calculador e o fim físico da RAM.

RUN também executa um CLEAR (limpa variáveis), porém não modifica a RAMTOP.

CÓDIGO DE MÁQUINA

25

25. Código de Máquina

O desenvolvimento de programas em código de máquina é um tanto mais elaborado e requer estudo apropriado. O capítulo é dirigido à programação avançada.

Neste capítulo serão vistos alguns conceitos básicos deste método eficiente de programação.

25.1. Código Binário

Relembrando o que foi visto no capítulo 19, o Z8Ø entende apenas sinais binários, linguagem de baixo nível, voltada para a máquina.

A elaboração de programas em formato binário é muito trabalhosa. Para facilitar o entendimento e o desenvolvimento de programas nesse formato, foram convencionados símbolos que representassem os binários de uma forma mais intelegível. O conjunto desses símbolos é chamado de linguagem Assembly.

Quando se trabalha com a linguagem BASIC, o computador encarrega-se de executar estas instruções de alto nível por intermédio de um interpretador.

Através de um programa utilitário (Monitor Assembler), pode-se desenvolver programas em linguagem de máquina, sem o auxílio do interpretador BASIC, tornando assim mais ágil o processamento.

Sendo concluído um programa em linguagem de máquina, o próximo passo será introduzi-lo no computador com o auxílio do Monitor Assembler, a partir de um endereço predeterminado.

Existem áreas na memória reservadas para uso do próprio sistema operacional do computador (veja o Mapa de Memória). Caso o endereço inicial especificado para o programa corresponda ao utilizado pelo Sistema Operacional, poderão ser produzidos resultados inesperados. Deve-se tomar um cuidado especial na definição do endereço inicial do programa em Assembly, que deve estar entre o BASIC e os caracteres definidos pelo usuário.

Pode-se trabalhar com o BASIC juntamente com rotinas criadas em linguagem de máquina. Para tanto deve-se definir a área a ser ocupada pela rotina em Assembly, atribuindo novo valor a RAMTOP. Exemplo:

```
CLEAR 32299 - (16K)
CLEAR 65Ø67 - (48K)
```

Isto fará com que o espaço reservado para esse fim tenha 3ØØ bytes. Lembrar que a área de definição de gráficos começa em 326ØØ (para 16 K) ou em 65368 (para 48 K).

Vamos ver um exemplo de como ficaria um programa utilizando o Monitor Assembler e também o mesmo programa em BASIC:

ASSEMBLER	BASIC
LD bc, 20	5 DATA 1,20,0,201 10 LET e = 32300 (ou 65068 p/48 K)
ret	15 FOR x = 1 TO 4 20 READ c: POKE e,c 25 LET e = e+1:NEXT x

O programa insere 20 no par de registradores bc.

Temos então em linguagem de máquina os respectivos valores: 01, 20, 00 e 201

01 - código de LD bc, (valor carregado no registrador c)
20 - valor carregado no registrador b
00 - valor carregado no registrador b
201 - código ret (retorno de sub-rotina em cod. maq.)

Utilizamos, em BASIC, o comando POKE, para inserir, a partir do endereço 32299 (ou 65067), os respectivos códigos.

25.2. USR Número

Rotinas em linguagem de máquina podem ser executadas num programa BASIC, usando-se a função USR. O argumento desta função é o endereço de início da sub-rotina, e seu resultado é um inteiro de dois bytes, sem sinal, ou com o sinal do par de registradores bc em retorno.

O endereço de retorno para o BASIC é mantido na forma normal, sendo o retorno efetuado através de uma instrução "ret".

Veja o exemplo:

PRINT USR 32300 (ou 65068)

O resultado apresentado será 20.

O "ret" é a instrução de retorno ao BASIC após a execução do programa em Código de Máquina, cujo endereço está na pilha de forma convencional.

Para armazenar em fita um programa em Código de Máquina, procede-se da seguinte forma:

SAVE "nome do programa" CODE end. inicial, quant. bytes

Para leitura:

LOAD "nome do programa" CODE end. inicial, quant. bytes

Use SAVE...LINE para a execução imediata após a leitura.

Para esclarecer melhor o tema acima abordado, sugerimos a consulta a literatura técnica sobre a Linguagem Assembly.

25.3. IN - OUT

As funções IN e OUT são utilizadas na programação avançada. De formato análogo às funções PEEK e POKE, IN e OUT desempenham um papel de leitor <IN> e determinador <OUT> dos valores que se encontram nas portas de I/O (entrada/saída).

Formato IN e OUT:

```
IN  endereço
OUT endereço, valor
```

Exemplos:
```
PRINT IN 200
OUT 16384,20
```

As portas de entrada e saída são usadas pelo processador para comunicar-se com periféricos adicionais e teclado. Estas portas podem ser controladas por IN e OUT para receber ou enviar dados destes dispositivos.

PERIFÉRICOS

26. PERIFÉRICOS

A saída de expansão do TK90X possibilita a conexão de vários periféricos e acessórios disponíveis no mercado.

SPEED-DRIVE

O speed-drive permite o armazenamento de grande quantidade de informações. É mais prático que um gravador cassete, pois facilita o acesso aos dados.

Este periférico trabalha com os comandos SAVE, VERIFY, LOAD, MERGE, PRINT, LIST, INPUT e INKEY$ além das palavras reservadas ao seu uso específico; OPEN #, CLOSE #, MOVE, ERASE, CAT e FORMAT.

IMPRESSORA

O TK90X pode ser acoplado a uma impressora paralela padrão CENTRONICS ou a uma impressora serial (conexão através da interface RS232-C).

Os comandos que atuam sobre este periférico são: LPRINT, LLIST, TAB, AT e COPY.

OUTROS

Além dos dois periféricos acima relacionados, podem ser conectados muitos outros, dentre eles:

Interface serial (RS232-C)

Interface paralela

Light-Pen

Cartuchos

Joystick (corresponde as teclas 6,7,8,9 e 0

EXPANSION

Conector que permite a passagem de 56 sinais, dispostos na seguinte ordem:

1 - A14
2 - A12
3 - 5 Volts
4 - 9 Volts
5 - guia
6 - GND
7 - não utilizado
8 - Clock
9 - A0
10 - A1
11 - A2
12 - A3
13 - $\overline{\text{IORQGE}}$
14 - GND
15 - GND
16 - não utilizado
17 - não utilizado
18 - não utilizado
19 - $\overline{\text{BUSREQ}}$
20 - $\overline{\text{RESET}}$
21 - A7
22 - A6
23 - A5
24 - A4
25 - $\overline{\text{ROMCS}}$
26 - $\overline{\text{BUSACK}}$
27 - A9
28 - A11

56 - A15
55 - A13
54 - D7
53 - SOUND IN
52 - guia
51 - D0
50 - D1
49 - D2
48 - D6
47 - D5
46 - D3
45 - D4
44 - $\overline{\text{INT}}$
43 - $\overline{\text{NMI}}$
42 - $\overline{\text{HALT}}$
41 - $\overline{\text{MREQ}}$
40 - $\overline{\text{IOREQ}}$
39 - $\overline{\text{RD}}$
38 - $\overline{\text{WR}}$
37 - - 5 Volts
36 - $\overline{\text{WAIT}}$
35 - 12 Volts
34 - 12 Volts
33 - $\overline{\text{M1}}$
32 - $\overline{\text{RFSH}}$
31 - A8
30 - A10
29 - não utilizado

APÊNDICE A

APÊNDICE A

PROGRAMAS:

Este apêndice destina-se a arrolar programas razoalvemente simples para demonstrar algumas habilidades do seu TK90X.

Paisagem

O exemplo é apropriado para se ter uma idéia da alta-resolução gráfica do seu microcomputador e da nitidez das cores que ele produz.

Sua execução gera a representação de um belo alvorecer marinho.

Sugestão: gravar apenas a tela com SAVE "nome" SCREEN$.

```
  5 PAPER 1: BORDER Ø: CLS
  6 GOSUB 12Ø
  8 INK Ø
 1Ø FOR c=1 TO 129 STEP 3
 15 PLOT Ø+c,8Ø
 2Ø DRAW 45-c,4Ø,-1
 25 PLOT 5Ø+c,8Ø
 3Ø DRAW 3Ø-c,2Ø,-1
 4Ø NEXT c
 65 FOR c=1 TO 48 STEP 2
 7Ø PLOT 178+c,8Ø
 75 DRAW 2Ø-c,8,-.Ø1
 8Ø PLOT 2Ø6+c,8Ø
 85 DRAW 1Ø-c,6,-1
 9Ø NEXT c
1ØØ FOR k=2 TO Ø.Ø1 STEP -.Ø5
1Ø5 PLOT INK 6;118,81
11Ø DRAW INK 6;6Ø,Ø,-k
115 NEXT k
117 STOP
12Ø FOR s=174 TO 85 STEP -1
125 PLOT INK 5;Ø,s
13Ø DRAW PAPER 5; INVERSE 1; INK Ø;255,Ø: DRAW PAPER 5; INVERSE 1; INK Ø;-255,-1
135 NEXT s
15Ø RETURN
```

Os programas a seguir não têm pretensões científicas. Eles são apenas ilustrativos, mas você pode aproveitar seus fundamentos para obter gráficos bastante úteis:

Curva do SIN

```
 5 BRIGHT 1: BORDER 5: CLS
10 PRINT AT 0,16;"**curva de sin**"
15 FOR X=0 TO 255
20 PLOT INK 1;x,87
25 PLOT INK 1;127,175-x
30 NEXT x
35 FOR m=-20 TO 20 STEP .10
40 PLOT INK 2;m*6+127,87+20*SIN m
45 NEXT m
```

Curvas SIN - COS - SQR

```
<NEW>-<ENTER>
 5 BRIGHT 1: BORDER 5: PAPER 7: CLS
10 INK 0
15 FOR v=-1 TO 254
20 PLOT 0,174-v: PLOT v,87.5
25 NEXT v
30 PRINT INK 2;AT 20,2;"sqr";AT 0,2;"cos";AT 9,2;"sin"
35 FOR n=5 TO 250
40 INK 4: PLOT n,90+80*SIN (n/128*PI)
45 INK 1: PLOT n,80*SQR (n/64)
50 INK 3: PLOT n,90+80*COS (n/128*PI)
55 NEXT n
```

MÚSICA

Green Leaves to a Ground

```
<NEW>-<ENTER>
 5 DIM n(75): DIM t(75)
10 DATA 2,5,7,9,10,9,7,4,0,2,4,5,2,2,1,2,4,1,-3,2,5,7,9,10,9,7,4,0,2,4,5,4,2,1,-1,1,2,69.84,12,12,12,11
11 DATA 9,7,4,0,2,4,5,2,2,1,2,4,1,-3,69.84,12,12,12,11,9,7,4,0,2,4,5,4,2,1,-1,1,2,69.84
15 DATA  .3,.6,.3,.6,.1,.1,.6,.3,.6,.1,.1,.6,.3,.6,.1,.1,.6,.3,.6,.3,.6,.3,.6,.1,.1,.6,.3,.6,.1,.1,.3,.3,.3,.3,.3,
   .3,.6.,6,.6,.3,.3
16 DATA .3,.3,.6,.3,.3,.3,.3,.6,.3,.3,.3,.3,.6,.3,.6,.6,.6,.3,.3,.3,.3,.6,.3,.3,.3,.3,.3,.3,.3,.3,.3,.3,1,.6
20 FOR x=1 TO 75: READ n(x): NEXT x
25 FOR x=1 TO 75: READ t(x): NEXT x
26 FOR a=1 TO 2
30 FOR s=1 TO 75: SOUND t(s),n(s): NEXT s
35 NEXT a
```

Digite e rode este programa, e divirta-se tentando adivinhar o que ele pede:

Adivinhe

```
<NEW>-<ENTER> <CAPS SHIFT><2>
  1 BORDER 1: PAPER 6
  2 CLS
  4 PRINT PAPER 2; INK 7;"###### ADIVINHE SE PUDER ######"
  5 LET N=INT (RND*100)
 10 FOR C=10 TO 1 STEP -1
 15 PRINT AT 8,4;"VOCE TEM    CHANCES PARA "
 16 PRINT AT 8,13; INK 0;c
 17 IF c=1 THEN SOUND .2,-10: SOUND .2,-10: SOUND .2,-10: SOUND 1,-14
 20 PRINT AT 9,5;"ADIVINHAR O NUMERO "; INK 2;"""N"""
 30 INPUT "DE O PALPITE (0 A 100): ";P
 35 IF P>N THEN PRINT AT 12,7;P;" E MAIOR QUE "; INK 1;"""N""";"    "
 40 IF P<N THEN PRINT AT 12,7;P;" E MENOR QUE "; INK 1;"""N""";"    "
 45 IF P=N THEN PRINT INK 2; FLASH 1;AT 12,4;"PARABENS!!! VOCE ACERTOU": GOTO 200
 46 SOUND .3,-12
 50 NEXT C
 60 IF C=0 AND N<>P THEN GOSUB 300
100 CLS
105 PRINT AT 20,3;"QUER JOGAR NOVAMENTE (S/N)"
106 PAUSE 20:FOR T=1 TO 60 STEP 10:SOUND. 1,T: NEXT T
110 IF INKEY$="" THEN GOTO 110
115 IF INKEY$="S" THEN GOTO 2
120 CLS: PRINT PAPER 4;"FIM": STOP
200 FOR T=1 TO 8
210 SOUND .1,10: SOUND .1,20: SOUND .1,30: NEXT T
220 GOTO 100
300 FOR T=-8 TO -20 STEP-1:SOUND .08,T:NEXT T
305 RETURN
```

APÊNDICE B

APÊNDICE B

MENSAGENS

O TK90X é dotado de um sistema que informa ao usuário, através de mensagens, qualquer situação que ocasione uma parada. A interrupção pode ser causada por um erro de programação, por um comando de parada do tipo BREAK ou STOP, ou pela simples conclusão da execução do programa.

Existem no sistema 27 mensagens precedidas por um código numérico de Ø a 9 ou uma letra de A a R. Além do código e do texto da mensagem, há também dois números separados por dois pontos. O primeiro número informa qual foi a última linha do programa que o computador executou. O segundo indica quantos comandos da última linha foram executados.

Por exemplo:

9 STOP Executado 3Ø:3

significa que o programa foi executado até a linha 3Ø e que parou após a terceira instrução ter sido efetuada. Cada mensagem tem característica relativa à situação que a gerou.

A seguir você tem uma lista de todas as mensagens, seu significado e a circunstância na qual a mensagem pode ocorrer.

Nota: X e Y serão usados para simbolizar os números que aparecem no final das mensagens separados por dois pontos.

Ø Executado X:Y

Término de uma execução bem sucedida ou desvio para um número de linha superior ao mais alto número de linha de um determinado programa.

1 NEXT sem FOR X:Y

A variável que controla o número de "loops" não foi definida pela instrução FOR, mas existe uma variável comum com o mesmo nome da que está no NEXT.

2 Variável inexistente X:Y

Tentativa de utilizar variável comum não definida no programa. A variável deveria ter sido introduzida por instrução de entrada de dados do tipo LET, READ ou INPUT, carregada de uma fita ou definida em um FOR. A mensagem ocorre também quando uma variável indexada foi utilizada antes de uma matriz ter sido dimensionada por DIM.

3 Erro de índice X:Y

Índice de variável fora das dimensões da matriz, indexação inadequada do dimensionamento de uma matriz.

4 Memória lotada X:Y

Tentativa de reservar ou ocupar espaço além do que a memória do computador dispõe. É necessário eliminar a linha que está sendo editada, caso o computador fique paralisado. Às vezes é preciso eliminar uma ou duas linhas de programa, utilizar o comando CLEAR, para obter mais espaço, e então reentrar as linhas eliminadas.

5 Excede área de vídeo X:Y

A mensagem ocorre quando há a tentativa de utilização da 23ª linha da tela, num comando do tipo PRINT AT 23,...

6 Número excede limite X:Y

Cálculo resultando em número maior que $1Ø^{38}$, aproximadamente.

7 RETURN sem GOSUB X:Y

Execução de um RETURN sem ter sido acessado por um GOSUB.

8 Fim de arquivo X:Y

Mensagem de speed-drive e outros periféricos.

9 STOP executado X:Y

Interrupção realizada pela presença de uma instrução STOP numa linha de programa. Para prosseguir a execução de eventuais linhas subseqüentes, deve ser usado o comando CONT.

A Argumento inválido X:Y

Introdução de um argumento que de alguma forma é impróprio para uma das seguintes funções: SQR, LN, ACS ou USR.

B Inteiro excede limite X:Y

Ocorre quando um inteiro é requisitado, e o valor introduzido (eventualmente arredondado) está aquém ou além da faixa permitida.

C Erro de sintaxe X:Y

O texto do argumento (string) não constitui expressão válida. O argumento está relacionado com VAL ou VAL$.

D BREAK-CONT repete X:Y

Acionamento de BREAK durante o funcionamento de algum periférico ou resposta negativa (N) ao scroll?. Se for usado CONT após a mensagem, o último comando será repetido.

E Fim de dados X:Y

Tentativa de usar o comando READ mais vezes do que permite o número de dados existentes em DATA.

F Nome inválido X:Y

Quando se usa SAVE e um nome de arquivo que excede 1Ø caracteres ou que não tem nenhum caractere (string vazio).

G Sem memória disponível

Não há mais lugar para acomodar uma nova linha de programa.

H STOP em INPUT X:Y

Introdução de STOP durante um INPUT. Se for usado CONT, o INPUT será repetido.

I FOR sem NEXT X:Y

Quando o valor determinado para a variável de controle do FOR não permitir que nenhum "loop" seja realizado, e também quando não foi encontrado o NEXT correspondente a tal variável. Por exemplo:

FOR x= 2 TO 1: INSTRUÇÕES: NEXT m

J Periférico inválido

Dispositivo de entrada/saída inválido.

K Cor inválida X:Y

O número escolhido está fora da faixa das cores existentes. Ocorre também quando se determina outros números que não Ø ou 1 para FLASH, BRIGHT, INVERSE ou OVER, ou após um dos caracteres de controle correspondentes.

L BREAK-CONT prossegue X:Y

A tecla BREAK foi pressionada durante a execução de um programa, entre dois comandos. Neste caso X e Y referem-se à linha e à instrução executadas antes do BREAK. Se CONT for acionado, o programa prosseguirá do ponto em que parou, sem repetir qualquer comando.

M RAMTOP inválido X:Y

Ocorre quando se determina um número muito grande ou muito pequeno para RAMTOP, através do comando CLEAR (às vezes com RUN também).

N Comando perdido X:Y

Tentativa de desvio para um comando inexistente. Ocorre geralmente com as instruções RETURN, NEXT e CONT.

O Canal inválido

Mensagem de speed-drive e outros periféricos.

P FN sem DEF X:Y

Quando há tentativa de se usar FN sem que tenha sido executado um DEF FN anteriormente.

Q Erro de parâmetro X:Y

Durante o uso de FN, a determinação dos argumentos foi incorreta quanto à quantidade ou ao tipo (string em lugar de número ou vice-versa).

R Erro de leitura X:Y

Por algum motivo, não foi possível ler os dados de uma fita.

APÊNDICE C

APÊNDICE C

VARIÁVEIS DO SISTEMA

NOTA	ENDEREÇO	NOME	CONTEÚDO
N8	23552	KBDWORK	Leitura do teclado.
N1	23560	KEYPRS	Acumula última tecla pressionada.
1	23561	RPTDLAY	Auto REPEAT, 1/50 de segundo para que uma tecla pressionada comece a repetir. Valor inicial: 35, podendo se alterar.
1	23562	RPTCCLE	1/50 de segundos de intervalo entre as repetições sucessivas de uma tecla pressionada. Valor inicial: 5.
N2	23563	PT DEF	Endereço dos argumentos de uma função definida pelo usuário. Valor inicial: 0.
N1	23565	K CLR	Acumula o segundo byte de controle de cor via teclado.
N2	23566	TVCLR	Acumula bytes de cor e controles AT e TAB enviados à TV.
X38	23568	PSTRM	Endereço dos canais ligados ao sistema.
2	23606	PTBLCHR	Menos 256 do endereço do conjunto de caracteres (começa com espaço e vai até o símbolo delta). Normalmente em ROM, mas pode ser definido em RAM e acessado por PTBLCHR.
1	23608	BUZCCLE	Intervalo do sinal sonoro.
1	23609	KCLICK	Intervalo do sinal sonoro do teclado.
1	23610	ERRCD	Inicia em 255 (para − 1), de modo que o conteúdo de PEEK 23610 é 255. Um a menos que o código da mensagem.
X1	23611	SFLAG0	Diversos FLAGS de controle do sistema BASIC.
X1	23612	SFLAG1	FLAGS associados à televisão.
X2	23613	P ERR	Endereço do elemento da pilha da máquina que deve ser usado como regresso do erro.
N2	23615	P LIST	Endereço de retorno da listagem automática.
N1	23617	CURSOR	Especifica cursor: **K**, **L**, **C**, **E** ou **G**.
2	23618	LNJMP	Linha-destino do salto.
1	23620	INSTRNR	Número da instrução na linha-destino do salto. Fazendo um POKE primeiramente para LNJMP e depois para INSTRNR, força um salto para uma instrução específica de uma linha.
2	23621	EXCLINE	Número de linha da instrução que está sendo executada.

NOTA	ENDEREÇO	NOME	CONTEÚDO
1	23623	SUBLEXC	Número dentro da linha da instrução que está sendo executada.
1	23624	BORCLR	Número da cor definida com BORDER vezes 8, mais os atributos da parte baixa da tela.
2	23625	CURLINE	Número da linha-corrente (com cursor do programa)
X2	23627	VARADD	Endereço das variáveis.
N2	23629	XVARADD	Endereço-destino da variável a ser modificada
X2	23631	CHCADD	Endereço dos dados do canal.
X2	23633	IOADD	Endereço da informação que está sendo utilizada para entrada/saída.
X2	23635	PROGBAS	Endereço do programa BASIC.
X2	23637	NEXEXC	Endereço da próxima linha do programa.
X2	23639	ENDDATA	Endereço do indicador do último elemento DATA.
X2	23641	INADD	Endereço da instrução a ser introduzida via teclado.
2	23643	CURADD	Endereço do cursor.
X2	23645	CHNXADD	Endereço do próximo caractere a ser interpretado: o caractere após o argumento PEEK, ou NEW LINE no fim da instrução POKE.
2	23647	SYCHADD	Endereço do caractere depois do ? .
X2	23649	WORKPT	Endereço do espaço de trabalho temporário.
X2	23651	STKCEND	Endereço do fim da pilha do calculador.
X2	23653	ADSPFREE	Endereço do início do espaço livre.
N1	23655	BREGCAL	Registro b do calculador.
N2	23656	MEMCADD	Endereço da área usada para a memória do calculador (geralmente MENSPCAL).
1	23658	SFLAG2	+ n FLAGS.
X1	23659	SIZE	Número de linhas de texto (incluindo-se uma em branco) na parte inferior da tela.
2	23660	LISTNR	Número inicial de linhas de um programa, para a listagem automática.
2	23662	CONTJMP	Número da linha-destino após CONT.
1	23664	CONTNR	Número de instruções dentro da linha-destino após CONT.

NOTA	ENDEREÇO	NOME	CONTEÚDO
N1	23665	SFLAG3	n FLAGS.
N2	23666	STRVRLEN	Comprimento da variável a ser modificada.
N2	23668	SYTADD	Endereço do próximo elemento da tabela de sintaxe.
2	23670	INITRND	O primeiro número para a instrução RND. Esta variável é inicializada por RAND.
3	23672	TVCOUNT	3 bytes, contador de imagens (o primeiro menos significativo). Incrementando todos os 20 ms.
2	23675	UDGRAPH	Endereço do primeiro caractere gráfico definido pelo usuário.
1	23677	LSTPLOT	Coordenada X do último ponto de um PLOT.
1	23678	COORDS	Coordenada Y do último ponto de um PLOT.
1	23679	POSIMPR	Número até 33 para posicionar coluna na impressora.
1	23680	PRTADD	Byte menos significativo do endereço da próxima posição em que LPRINT deve escrever (no buffer da impressora).
1	23681		Não usado.
2	23682	HVBFFIN	Número de colunas (até 33) e linhas (até 24) da memória de entrada.
2	23684	DFPSPRT	Endereço no Arquivo de Imagem da posição PRINT.
2	23686	DFPSPRTL	A mesma função de DFPSPRT, mas para a parte inferior do vídeo.
X1	23688	HV POS	Número de colunas (até 33) para a posição de PRINT.
X1	23689		Número de linhas (até 24) para a posição de PRINT.
X2	23690	HV POSL	Como S POSN, mas para a parte inferior.
1	23692	SCRINC	Contador para scroll: é sempre 1 a menos do que o número de scrolls que serão feitos antes da pergunta scroll?. Introduzindo-se continuamente neste endereço um número maior que 1, a tela continuará a fazer scroll ininterruptamente.
1	23693	ATCLR P	Situação de posição permanente de tela quanto à cor etc.
1	23694	MASKCLRP	Usado para cores transparentes, etc. Mostra que o bit de atributo corresponde não é pegado do ATCLR P, mas daquele que já está na tela.
N1	23695	ATCLR T	Situação de uma posição de tela
N1	23696	MASKCLRT	O mesmo que MASKCLRP, mas temporário.
1	23697	SFLAG4	n FLAG.

NOTA	ENDEREÇO	NOME	CONTEÚDO
N30	23698	MEMSPCAL	Área de memória do calculador usada para guardar números que não podem ser colocados convenientemente na pilha do calculador.
2	23728	NMIVCT	Usado na interrupção NMI.
2	23730	RAMTOP	Endereço do último byte da área do sistema BASIC.
2	23732	MEMAVLB	Endereço do último byte da RAM física.

APÊNDICE D

CÓDIGO	CARACTERE	HEXA	ASSEMBLY Z80	DEPOIS DE CB	DEPOIS DE ED
91	[	5B	Ld e,e	bit 3,e	Ld de,(NN)
92	/	5C	Ld e,h	bit 3,h	
93	]	5D	Ld e,l	bit 3,l	
94	↑	5E	Ld e,(hl)	bit 3,(hl)	im 2
95	–	5F	Ld e,a	bit 3,a	Ld a,r
96	Σ	60	Ld h,b	bit 4,b	in h,(c)
97	a	61	Ld h,c	bit 4,c	out (c),h
98	b	62	Ld h,d	bit 4,d	sbc hl,hl
99	c	63	Ld h,e	bit 4,e	Ld (NN),hl
100	d	64	Ld h,h	bit 4,h	
101	e	65	Ld h,l	bit 4,l	
102	f	66	Ld h,(hl)	bit 4,(hl)	
103	g	67	Ld h,a	bit 4,a	rrd
104	h	68	Ld l,b	bit 5,b	in l,(c)
105	i	69	Ld l,c	bit 5,c	out (c),l
106	j	6A	Ld l,d	bit 5,d	adc hl,hl
107	k	6B	Ld l,e	bit 5,e	Ld hl,(NN)
108	l	6C	Ld l,h	bit 5,h	
109	m	6D	Ld l,l	bit 5,l	
110	n	6E	Ld l,(hl)	bit 5,(hl)	
111	o	6F	Ld l,a	bit 5,a	rld
112	p	70	Ld (hl),b	bit 6,b	in f,(c)
113	q	71	Ld (hl),c	bit 6,c	
114	r	72	Ld (hl),d	bit 6,d	sbc hl,sp
115	s	73	Ld (hl),e	bit 6,e	Ld (NN),sp
116	t	74	Ld (hl),h	bit 6,h	
117	u	75	Ld (hl),l	bit 6,l	
118	v	76	halt	bit 6,(hl)	
119	w	77	Ld (hl),a	bit 6,a	
120	x	78	Ld a,b	bit 7,b	in a,(c)
121	y	79	Ld a,c	bit 7,c	out (c),a
122	z	7A	Ld a,d	bit 7,d	adc hl,sp
123	{	7B	Ld a,e	bit 7,e	Ld sp,(NN)
124	¦	7C	Ld a,h	bit 7,h	
125	}	7D	Ld a,l	bit 7,l	
126	~	7E	Ld a,(hl)	bit 7,(hl)	
127	Δ	7F	Ld a,a	bit 7,a	
128	□	80	add a,b	res 0,b	
129	▝	81	add a,c	res 0,c	
130	▘	82	add a,d	res 0,d	
131	▀	83	add a,e	res 0,e	
132	▗	84	add a,h	res 0,h	
133	▐	85	add a,l	res 0,l	
134	▚	86	add a,(hl)	res 0,(hl)	
135	▜	87	add a,a	res 0,a	
136	▖	88	adc a,b	res 1,b	
137	▞	89	adc a,c	res 1,c	
138	▌	8A	adc a,d	res 1,d	

CÓDIGO	CARACTERE	HEXA	ASSEMBLY Z80	DEPOIS DE CB	DEPOIS DE ED
139	▛	8B	adc a,e	res 1,e	
140	▄	8C	adc a,h	res 1,h	
141	▟	8D	adc a,l	res 1,l	
142	▙	8E	adc a,(hl)	res 1,(hl)	
143	█	8F	adc a,a	res 1,a	
144	(a) UDG	90	sub b	res 2,b	
145	(b) UDG	91	sub c	res 2,c	
146	(c) UDG	92	sub d	res 2,d	
147	(d) UDG	93	sub e	res 2,e	
148	(e) UDG	94	sub h	res 2,h	
149	(f) UDG	95	sub l	res 2,l	
150	(g) UDG	96	sub (hl)	res 2,(hl)	
151	(h) UDG	97	sub a	res 2,a	
152	(i) UDG	98	sbc a,b	res 3,b	
153	(j) UDG	99	sbc a,c	res 3,c	
154	(k) UDG	9A	sbc a,d	res 3,d	
155	(l) UDG	9B	sbc a,e	res 3,e	
156	(m) UDG	9C	sbc a,h	res 3,h	
157	(n) UDG	9D	sbc a,l	res 3,l	
158	(o) UDG	9E	sbc a,(hl)	res 3,(hl)	
159	(p) UDG	9F	sbc a,a	res 3,a	
160	(q) UDG	A0	and b	res 4,b	Ldi
161	(r) UDG	A1	and c	res 4,c	cpi
162	(s) UDG	A2	and d	res 4,d	ini
163	(t) UDG	A3	and e	res 4,e	outi
164	(u) UDG	A4	and h	res 4,h	
165	RND	A5	and l	res 4,l	
166	INKEY$	A6	and (hl)	res 4,(hl)	
167	PI	A7	and a	res 4,a	
168	FN	A8	xor b	res 5,b	Ldd
169	POINT	A9	xor c	res 5,c	cpd
170	SCREEN$	AA	xor d	res 5,d	ind
171	ATTR	AB	xor e	res 5,e	outd
172	AT	AC	xor h	res 5,h	
173	TAB	AD	xor l	res 5,l	
174	VAL$	AE	xor (hl)	res 5,(hl)	
175	CODE	AF	xor a	res 5,a	
176	VAL	B0	or b	res 6,b	Ldir
177	LEN	B1	or c	res 6,c	cpir
178	SIN	B2	or d	res 6,d	inir
179	COS	B3	or e	res 6,e	otir
180	TAN	B4	or h	res 6,h	
181	ASN	B5	or l	res 6,l	
182	ACS	B6	or (hl)	res 6,(hl)	
183	ATN	B7	or a	res 6,a	
184	LN	B8	cp b	res 7,b	Lddr
185	EXP	B9	cp c	res 7,c	cpdr
186	INT	BA	cp d	res 7,d	indr

VARIÁVEIS DO SISTEMA

CÓDIGO	CARACTERE	HEXA	ASSEMBLY Z80	DEPOIS DE CB	DEPOIS DE ED
187	SQR	BB	cp e	res 7,e	otdr
188	SGN	BC	cp h	res 7,h	
189	ABS	BD	cp l	res 7,l	
190	PEEK	BE	cp (hl)	res 7,(hl)	
191	IN	BF	cp a	res 7,a	
192	USR	C0	ret nz	set 0,b	
193	STR$	C1	pop bc	set 0,c	
194	CHR$	C2	jp nz,NN	set 0,d	
195	NOT	C3	jp NN	set 0,e	
196	BIN	C4	call nz,NN	set 0,h	
197	OR	C5	push bc	set 0,l	
198	AND	C6	add a,N	set 0,(hl)	
199	<=	C7	rst 0	set 0,a	
200	>=	C8	ret z	set 1,b	
201	<>	C9	ret	set 1,c	
202	LINE	CA	jp z,NN	set 1,d	
203	THEN	CB		set 1,e	
204	TO	CC	call z,NN	set 1,h	
205	STEP	CD	call NN	set 1,l	
206	DEF FN	CE	adc a,N	set 1,(hl)	
207	CAT	CF	rst 8	set 1,a	
208	FORMAT	D0	ret nc	set 2,b	
209	MOVE	D1	pop de	set 2,c	
210	ERASE	D2	jp nc,NN	set 2,d	
211	OPEN #	D3	out (n),a	set 2,e	
212	CLOSE #	D4	call nc,NN	set 2,h	
213	MERGE	D5	push de	set 2,l	
214	VERIFY	D6	sub N	set 2,(hl)	
215	SOUND	D7	rst 16	set 2,a	
216	CIRCLE	D8	ret c	set 3,b	
217	INK	D9	exx	set 3,c	
218	PAPER	DA	jp c,NN	set 3,d	
219	FLASH	DB	in a,(N)	set 3,e	
220	BRIGHT	DC	call c,NN	set 3,h	
221	INVERSE	DD	util. inst. prefixos ix	set 3,l	
222	OVER	DE	sbc a,N	set 3,(hl)	
223	OUT	DF	rst 24	set 3,a	
224	LPRINT	E0	ret po	set 4,b	
225	LLIST	E1	pop hl	set 4,c	
226	STOP	E2	jp po,NN	set 4,d	
227	READ	E3	ex(sp),hl	set 4,e	
228	DATA	E4	call po,NN	set 4,h	
229	RESTORE	E5	push hl	set 4,l	
230	NEW	E6	and N	set 4,(hl)	
231	BORDER	E7	rst 32	set 4,a	
232	CONT	E8	ret pe	set 5,b	
233	DIM	E9	jp (hl)	set 5,c	
234	REM	EA	jp pe,NN	set 5,d	

CÓDIGO	CARACTERE	HEXA	ASSEMBLY Z80	DEPOIS DE CB	DEPOIS DE ED
235	FOR	EB	ex de,hl	set 5,e	
236	GOTO	EC	call pe,NN	set 5,h	
237	GOSUB	ED		set 5,l	
238	INPUT	EE	xor N	set 5,(hl)	
239	LOAD	EF	rst 40	set 5,a	
240	LIST	F0	ret p	set 6,b	
241	LET	F1	pop af	set 6,c	
242	PAUSE	F2	jp p,NN	set 6,d	
243	NEXT	F3	di	set 6,e	
244	POKE	F4	call p,NN	set 6,h	
245	PRINT	F5	push af	set 6,l	
246	PLOT	F6	or N	set 6,(hl)	
247	RUN	F7	rst 48	set 6,a	
248	SAVE	F8	ret m	set 7,b	
249	RAND	F9	ld sp,hl	set 7,c	
250	IF	FA	jp m,NN	set 7,d	
251	CLS	FB	ei	set 7,e	
252	DRAW	FC	call m,NN	set 7,h	
253	CLEAR	FD	util. inst. prefixos iy	set 7,l	
254	RETURN	FE	cp N	set 7,(hl)	
255	COPY	FF	rst 56	set 7,a	

APÊNDICE D

VARIÁVEIS DO SISTEMA

CÓDIGO	CARACTERE	HEXA	ASSEMBLY Z80	DEPOIS DE CB	DEPOIS DE ED
0	UDG	00	nop	rlc b	
1	TRACE	01	Ld bc,N	rlc c	
2	não usado	02	Ld (bc),a	rlc d	
3	não usado	03	inc bc	rlc e	
4	não usado	04	inc b	rlc h	
5	não usado	05	dec b	rlc l	
6	PRINT vírgula	06	Ld, b, N	rlc (hl)	
7	EDIT	07	rlca	rlc a	
8	cursor para esquerda	08	ex af,af'	rrc b	
9	cursor para direita	09	add hl,bc	rrc c	
10	cursor em baixo	0A	Ld a,(bc)	rrc d	
11	cursor em cima	0B	dec bc	rrc e	
12	DELETE	0C	inc c	rrc h	
13	ENTER	0D	dec c	rrc l	
14	Número	0E	Ld c,N	rrc (hl)	
15	Não usado	0F	rrca	rrc a	
16	INK control	10	djnz DIS	rl b	
17	PAPER control	11	Ld de,NN	rl c	
18	FLASH control	12	Ld (de),a	rl d	
19	BRIGHT control	13	inc de	rl e	
20	INVERSE control	14	inc d	rl h	
21	OVER control	15	dec d	rl l	
22	AT control	16	Ld d, N	rl (hl)	
23	TAB control	17	rla	rl a	
24	não usado	18	jr DIS	rr b	
25	não usado	19	add hl,de	rr c	
26	não usado	1A	Ld a,(de)	rr d	
27	não usado	1B	dec de	rr e	
28	não usado	1C	inc e	rr h	
29	não usado	1D	dec e	rr l	
30	não usado	1E	Ld e,N	rr (hl)	
31	não usado	1F	rra	rr a	
32	Espaço	20	jr nz,DIS	sla b	
33	!	21	Ld hl,NN	sla c	
34	"	22	Ld (NN),hl	sla d	
35	#	23	inc hl	sla e	
36	$	24	inc h	sla h	
37	%	25	dec h	sla l	
38	&	26	Ld h,N	sla (hl)	
39	'	27	daa	sla a	
40	(	28	jr z,DIS	sra b	
41	)	29	add hl,hl	sra c	
42	*	2A	Ld hl,(NN)	sra d	

CÓDIGO	CARACTERE	HEXA	ASSEMBLY Z80	DEPOIS DE CB	DEPOIS DE ED
43	+	2B	dec hl	sra e	
44	,	2C	inc l	sra h	
45	–	2D	dec l	sra l	
46	.	2E	Ld l,N	sra (hl)	
47	/	2F	cpl	sra a	
48	0	30	jr nc,DIS		
49	1	31	Ld sp,NN		
50	2	32	Ld (NN),a		
51	3	33	inc sp		
52	4	34	inc (hl)		
53	5	35	dec (hl)		
54	6	36	Ld (hl),N		
55	7	37	scf		
56	8	38	jr c,DIS	srl b	
57	9	39	add hl,sp	srl c	
58	:	3A	Ld a,(NN)	srl d	
59	;	3B	dec sp	srl e	
60	<	3C	inc a	srl h	
61	=	3D	dec a	srl l	
62	>	3E	Ld a,N	srl (hl)	
63	?	3F	ccf	srl a	
64	@	40	Ld b,b	bit 0,b	in b, (c)
65	A	41	Ld b,c	bit 0,c	out (c),b
66	B	42	Ld b,d	bit 0,d	sbc hl,bc
67	C	43	Ld b,e	bit 0,e	Ld (NN),bc
68	D	44	Ld b,h	bit 0,h	neg
69	E	45	Ld b,l	bit 0,l	retn
70	F	46	Ld b,(hl)	bit 0,(hl)	im 0
71	G	47	Ld b,a	bit 0,a	Ld i,a
72	H	48	Ld c,b	bit 1,b	in c,(c)
73	I	49	Ld c,c	bit 1,c	out (c),c
74	J	4A	Ld c,d	bit 1,d	adc hl,bc
75	K	4B	Ld c,e	bit 1,e	Ld b,c(NN)
76	L	4C	Ld c,h	bit 1,h	
77	M	4D	Ld c,l	bit 1,l	reti
78	N	4E	Ld c,(hl)	bit 1,(hl)	
79	O	4F	Ld c,a	bit 1,a	Ld r,a
80	P	50	Ld d,b	bit 2,b	in d,(c)
81	Q	51	Ld d,c	bit 2,c	out (c),d
82	R	52	Ld d,d	bit 2,d	sbc hl,de
83	S	53	Ld d,e	bit 2,e	Ld (NN),de
84	T	54	Ld d,h	bit 2,h	
85	U	55	Ld d,l	bit 2,l	
86	V	56	Ld d,(hl)	bit 2,(hl)	im 1
87	W	57	Ld d,a	bit 2,a	Ld a,i
88	X	58	Ld e,b	bit 3,b	in e,(c)
89	Y	59	Ld e,c	bit 3,c	out (c),e
90	Z	5A	Ld e,d	bit 3,d	adc hl,de

SUGESTÕES DO USUÁRIO

Toda e qualquer sugestão ou observação que você queira fazer, no sentido de melhorar este manual, será bem recebida e de grande valia para as futuras edições.

....

....

....

....

....

....

....

....

....

....

....

Nome: ....

Endereço:.. Tel.:...

CEP: .... Cidade: .... Estado: .... País: ....

Favor enviar para:

Microdigital Eletrônica LTDA.
Departamento de Publicações Técnicas
Caixa Postal n.º 54.121
Ø1296 - São Paulo (SP)
Brasil

CERTIFICADO DE GARANTIA

Este produto é garantido pela Microdigital Eletrônica Ltda., de acordo com as seguintes normas:

1) A Microdigital assegura, ao primeiro comprador-usuário deste produto, garantia de 90 (noventa) dias a partir da data de emissão da nota fiscal de venda, emitida pelo Revendedor Microdigital, contra qualquer defeito de peças ou fabricação, desde que seja constatada falha ou defeito em condições normais de uso por técnico da Microdigital ou devidamente autorizado pela mesma.

2) Excluem-se desta garantia: fonte, circuitos integrados, cabos de vídeo, cristal, semicondutores.

3) A execução dos serviços, durante o período da garantia sem ônus para o comprador-usuário, fica restrita ao território brasileiro, onde a Microdigital mantiver Unidade de Serviço Técnico, próprias ou autorizadas.

4) Nas demais localidades onde a Microdigital não possuir Unidade de Serviço Técnico, as despesas de transportes e seguro do aparelho ida e volta, correrão por conta do comprador-usuário.

5) Esta garantia só terá validade se acompanhada da respectiva Nota Fiscal de venda, e preenchida de modo legível.

ESTA GARANTIA PERDERÁ SUA VALIDADE CASO:

A) O prazo de 90 (noventa) dias seja ultrapassado

B) Uso indevido ou em desacordo com o Manual de Operação

C) Ligação seja feita em rede elétrica imprópria ou sujeita a flutuações

D) Haja danos causados por agentes da Natureza

E) Ocorra acidente com o aparelho

F) Seja verificado violação de lacre do aparelho ou abertura do mesmo por técnico não autorizado pela Microdigital

G) Não estiver acompanhado da nota fiscal de venda

Nome do Comprador ______________________________

Endereço ______________________________

Cidade ______________________ Estado __________ CEP __________

Revendedor ______________________________

Endereço ______________________ Cidade ______________ Estado __________

Nota Fiscal n.º ______________ Data ___/___/___

Certificado de Garantia n.º

MICRODIGITAL ELETRÔNICA LTDA
Dpto. Suporte ao Usuário
Av. Angélica, 2318 - Mezanino - PABX: 255-0366
CEP 01228 - São Paulo - Brasil